VIC rapport

ÅRGÅNG 3 NR 9
PRIS 15:– inkl moms

PRESENTATION AV TÄVLINGS-RESULTAT

TEMA! KRINGUTRUSTNING



473-9

VIC kassettservice

VIC rapport erbjuder sina läsare följande service:

Alla längre program som publiceras i VIC rapport kommer att finnas till förfogande på en kassett, som du kan prenumerera på.

Pris	Prenum.	Icke Prenum.
3 nummer/kassetter	100:–	125:–
6 nummer/kassetter	190:–	215:–
10 nummer/kassetter	300:–	325:–



Om du är intresserad är det bara att fylla i nedanstående blankett och skicka in den till VIC rapport. Kassetten kommer sedan direkt hem i brevlådan.

Blanketten skickas till VIC rapport, Box 42054, 126 12 Stockholm.

Min prenumerantkod är .... och jag beställer en prenumeration på

- ☐ 3 kassetter för 100:–
- ☐ 6 kassetter för 190:–
- ☐ 10 kassetter för 300:–

Jag är inte prenumerant och beställer en prenumeration på

- ☐ 3 kassetter för 125:–
- ☐ 6 kassetter för 215:–
- ☐ 10 kassetter för 325:–

Namn: ______________________________

Adress: ______________________________

Postadress: ______________________________

Telefon: ______________________________

VIC rapport

INNEHÅLL NR. 9 1984

VIC redaktion: Box 42054, 126 12 Stockholm. Telefon 08-744 59 20
Ansvarig utgivare: Nina Linander
Administrativ redaktör: Elisabeth Höglund
Teknisk redaktör: Matts Nilsson
Omslagsteckning: Per Lindfors
Layout: Ordbyggarna
Övriga medarbetare: Åke Hedman, Joakim Aspengren, Ola Johansson, Åke Fredriksson, Bengt Litnäs, Ola Axelsson
Annonser: Marknadsmedia AB. Telefon 08-753 00 20
Tryck: Per-Bo AB, Stockholm 1984
ISSN-NR: 0281–8043
Annonsorder och annonsmaterial (hel- original eller negativ film) enligt överenskommelse.
Tryckförfarande: Offset.
Upplaga 20 000 ex.
Nr 10 av VIC rapport utkommer den 12 november. Manusstopp för nr 11/12 är den 19 oktober och annonsstopp den 12 november.

Nya produkter

Sitter och funderar på hur jag ska framföra följande till dig, läsare. Jag har nämligen fått klagomål. Klagomål på att VIC rapport skriver om produkter som inte finns och kanske aldrig kommer att finnas.

Jag vill därför påpeka att VIC rapport liksom andra tidningar rapporterar om nya produkter som visas på mässor, utställningar etc. Men även om en produkt visas på en mässa i London, Hannover eller något annat ställe, finns det ingen garanti för att den kommer ut eller lanseras i Sverige. VIC rapport lovar att rapportera när de verkligen är här och finns att köpa. Då bjuder vi även på en produkttest. Men tills dess kan vi, liksom andra datortidningar, bara rapportera det vi hör och läser om.

Just nu surrar det många rykten på hårdvarusidan. Ringde därför Handic Electronic för att ta reda på vad som händer på den svenska marknaden. De har fortfarande inte sett röken av de nya maskinerna från Commodore. Det finns enligt dem ingen garanti för när, hur och om de nya maskinerna kommer att lanseras. Tidigast under 1985 tror de.

Däremot såg det ljust ut för färgskrivaren MCS–801. Den kommer troligen ut på den svenska marknaden senare i höst. Observera troligen!!!

VIC rapport kommer fr o m nästa nummer att starta en underrättelsehörna i vilken vi rapporterar senaste nytt från Commodore och Handic Electronic. På så sätt får du löpande rapporter om vilka nya produkter som kommer, när och var.



Västtysk hårdvara för Commodore

I Västtyskland går utvecklingen av datorer, mjuk- och hårdvara snabbt framåt. Genom dollarkursens f n mycket oförmånliga ställning blir amerikanskt dyrt. Med en Mark-kurs på ca 3 kr kan inköp göras förmånligt.

Jag roade med med att titta i ELFA-katalogen på priserna på processorer och fann, att priserna i Västtyskland ligger på mellan 30% (Z 80) och 50% (6502) av de svenska priserna.

Vi skall då och då ta fram västtyska produkter och presentera dem för våra läsare. Kanske finner du något av intresse. Ofta finns ej återförsäljare i Sverige utan du får importera direkt. Tänk på att 24% moms tillkommer i Sverige och att du får betala mellan 15–40 kr i porto för lättare prylar.

EXPANDERINGSKORT TILL VIC 64 HIRNY

Insticksmodul för fem expanderingar av vilka tre är omkopplingsbara. Extern matarspänning kan anslutas 5V över en inbyggd spänningsregulator (max 1A). Likriktare är inbyggd på kortet.

Modulen har vidare ett uttag för att "knacka" moduler, den har vidare en optisk driftskontroll, RESET-knapp samt en integrerad miniatyrsäkring.

En beskrivning hur man går tillväga för att kopiera ett 8 KBytes ROM bifogas.

NÅGOT FÖR SKJUTGLADA

Quickfinger skall öka din träffsäkerhet med minst 40%. Slut på allt tryckande på knappar och vridande på spakar! Med två potentiometrar (vridmotstånd) kan du styra puls–paus-förhållandet och du får fram din egen personliga eldgivningsfrekvens. Den låter sig avläsas på en LED.

Quickfinger kopplas mellan kontrollutgången och Joystick. Prylen finns att köpa som byggsats. Vill du bespara dig ömma fingrar eller senskideinflammationer, så är detta ett måste!

CENTRONICS GRÄNSSNITT VIC 64

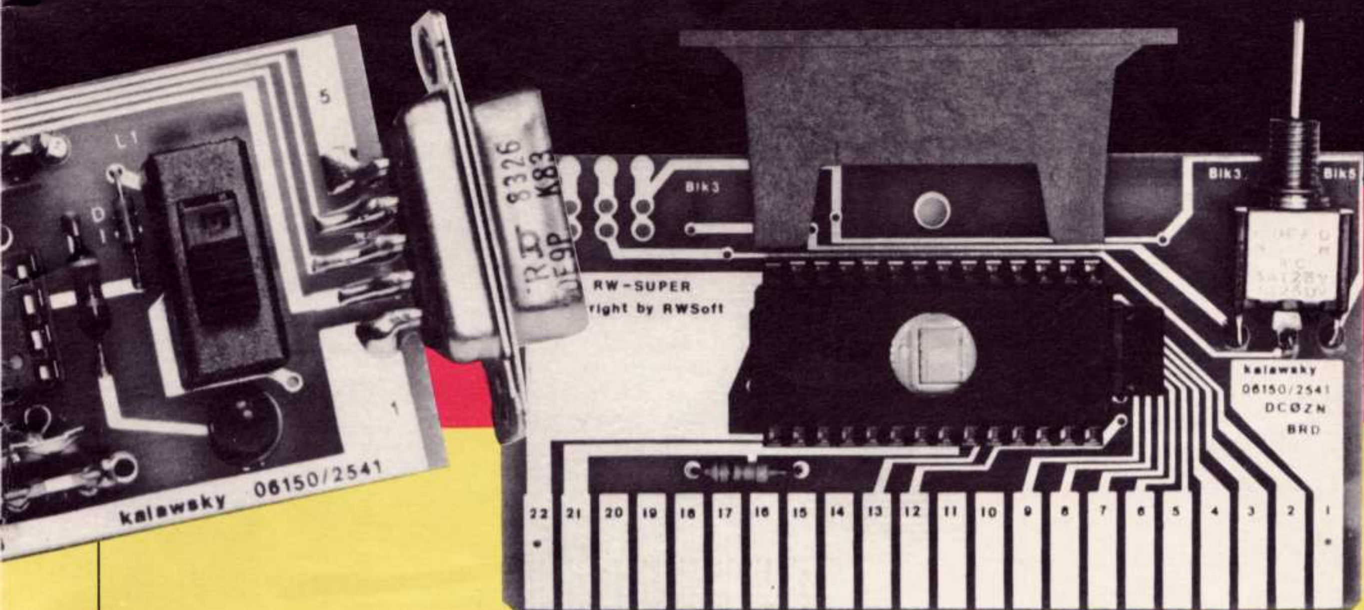
För att göra 64an kompatibel med andra datorer med seriell dataöverföring behövs detta gränssnitt. Normal 8-bitars överföring och skrivarens alla egenskaper bibehålles. Man kan erhålla hardcopy i två versioner
normal (hos Shinwa CP-80 ca 61* 94 mm ca 40 sek.)
dubbel (hos Shinwa CP-80 ca 122*188 mm ca 150 sek.)

Detta gränssnitt är ursprungligen avsett för textbearbetning men har även några andra egenskaper. Hardcopy av bildskärmen erhålles genom CTRL-F normal eller CTRL-X dubbelt eller man kan programmera det. Text och grafik kan fås med hardcopy, men ej sprites. Listningar utföres ej med Commodores specialtecken. Dessa kan tas ut som hardcopy. Gränssnittet levereras som en in- och urkopplingsbar insticksmodul.

64 KBYTE RAM-MASTER

Ett 64-KBkort avsett för VIC 20/64. Det är för VIC 20 omkopplingsbart inom alla minnesblocken t ex även 3 KByte. Kopplingen sker genom mjukvara (banking).

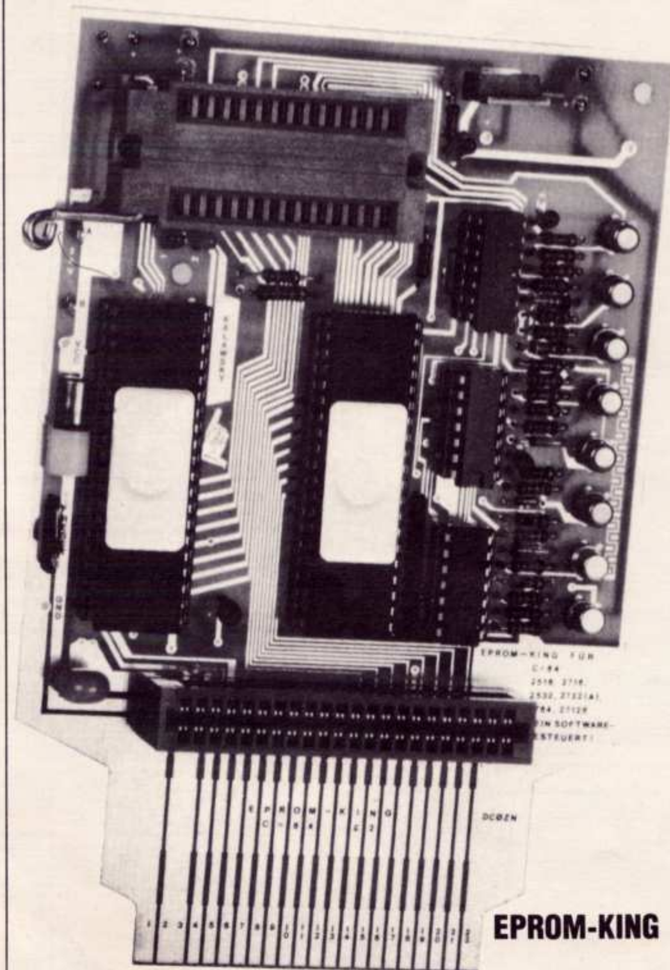
★ Minnesarea för ROM-moduler.
★ Mjukvarukompatibel med alla program.
★ Varje expansionssteg kan för sig in- och urkopplas.
★ Koppling för skrivskydd inbyggt.
★ Finns som byggsats eller färdig.



CENTRONICS GRÄNSSNITT

EPROM-KING FÖR VIC 20/64

★ Programmeringsapparat för följande 2–16 KB EPROMs: 2516, 2716, 2532, 2732, 2732A, 2764, 27128.
★ Styrs helt av mjukvara (ej kopplingar).
★ Inläsning för 8 K-ROMs möjlig.
★ Automatisk VERIFY-prövning efter programmeringen.
★ Integrerad Byte-Wide koncept.
★ Möjligt att kopiera även olika EPROM-typer.



EPROM-KING

★ Extra moduluttag.
★ Extern driftspänning (datorn belastas ej).
★ Snabbväxlingssocklar (0-socklar).
★ Användarvänlig mjukvara (på tyska).

Forts på sidan 6

Forts från sidan 5
"Västtysk hårdvara..."

40/80 KORT

★ Ständig information i färg på bildskärmen.
★ Delprogrammering implementerad.
★ Intelligent programalgoritm (från 2864 ca 1 min bränntid).
★ Driftsmjukvara kan fås på kassett, diskett el 8K-EPROM.
★ Finns endast som färdigprodukt.

40/80-TECKEN-KORT FÖR VIC 20

★ Valfritt 40*25 eller 80*25 tecken/rad.
★ Definition av statusrader.
★ Radavståndet kan förändras.
★ Bildskärmen kan rullas nedåt.
★ Ur- och inkoppling över mjukvara.
★ Vid samtidig användning av RAM MASTER-kortet är 31743 Bytes fria.
★ Över mjukvara kan 0,5 KB extra RAM inkopplas.
★ Genom att anskaffa ett modifierat driftsystem fungerar 40/80-kortet tillsammans med autostartade ROM-moduler.
★ Dubbla driftsystem varigenom andra program ej störtas.

RW-SUPER FÖR VIC 20 ROM-MODUL

★ Toolkitkommandon.
★ Snabba lagringar på kassett (ca 10×).
★ Maskinspråksmonitor inkl assembler/disassembler och debugger.
★ Korta kommandon för driven.
★ Centronics gränssnitt vid USER-utgången för styrning av skrivare.
★ Integrerad snabbladdare för ROM.
★ Universellt användbar omräknare HEX-DEC-BINÄR.
★ m m, m fl, o s v.

RW-HJÄLP FÖR VIC 64

Motsvarar i stort ovanstående för VIC 20.

BUSFÖRLÄNGNINGAR FÖR VIC 20/64

★ Kan erhållas för USER och expansionsutgång.
★ Inbyggd RESET-knapp.

REALTIDUR VIC 20/64

★ Batterimatat kvartsstyrt ur.
★ Tim, min, sek, mån, dag, år (skottårskorrigering)
★ Drivs över USER-utgången
★ Mjukvara på diskett
★ Olika sifferstorlekar kan göras.

Tillbehör från:
Ing.büro H. Kalansky
D 6108 Weiterstadt 1.

RAM-MASTER 64 KB

Nyheter för VICarna

En ny diskdrive är på gång och den heter SFD 1001/1002. Egentligen är det två. Commodore har delat på en 8250-drive och därmed skapat den nya.

Skillnaden mellan 1001 och 1002 är, att 1001 har en extra kabel för ca 400 kr med vilken man förbinder VIC 64 med ett gränssnitt IEEE–488.

Efter att ha formatterat en diskett meddelar 1001, att den har 4133 fria block att bjuda på, vilket är lika med 1003,25 KByte. På 77 spår lagras data på mellan 23 och 29 sektorer per spår. Överföringen sker med en hastighet av 1,2 KByte/sek. För VC 1541 är hastigheten 0,4 KByte.

Någon förbindelse mellan VC 1541 och 1001 är ej omedelbart för handen. Den seriella bussen är spärrad och kommunikation vare sig med VC 1541 eller med en skrivare är omöjlig.

Ett program som åtgärdar denna felaktighet finns emellertid och kommer säkerligen att ingå i demonstrationsdisketten eller som en listning att skriva själv.

1001 fordrar också en tillsatskabel för andra periferienheter såsom skrivare. Snittstället har på ovansidan ett uttag för moduler. Här har ytterligare ett fel uppstått: sätt i en modul och ingenting fungerar längre! För en pryl i 8 000 kronorsklassen kunde man kanske begära mer. (OBS! Produkten kommer till Sverige först under 1985.)

YTTERLIGARE EN DISKDRIVE FRÅN KONKURRENT

Denna drive är vertikalt stående och heter YL–55S1L–CM med ett drivverk från TEAC. Detta betyder kort och gott – driftsäkerhet. Tyvärr är denna drive ej snabbare än VC 1541 då Commodores DOS ej medger snabbare bearbetning. Firman lovar dock, att när man har löst problemet (arbetet pågår) så skall köpare av den nya driven få byta ut den gamla mot en ny till ett billigt pris. I Västtyskland kostar driven DM 800 = ca 2 300 kronor.

NY KASSETTBANDSPELARE

Från samma firma kommer också en ny kassettbandspelare för DM 99.–. För styrning finns 6 tangenter.

TELEFONMODEM SOM BYGGSATS

I Västtyskland erbjudes en byggsats för ett modem för ca DM 260.–. Satsen innehåller grundplatta och ca 200 separata delar måste lödas ihop innan det förhoppningsvis fungerar. □



UTNYTTJA DIN 64a SOM ETT MÄTINSTRUMENT

Alla som någon gång sysslat med elektronik har förmodligen snart upptäckt att det saknas användbara instrument när man som bäst behöver dem. Det är lagen om alltings j-het, som inverkar där. Själv har jag ett gammalt trött oscilloscop, som knappast duger till någonting, utom att hjälpligt kontrollera vissa vågformer. Moderna 'skåp' kostar en smärre förmögenhet. Gamla brukar vara 'synkade' och därmed odugliga till att mäta exempelvis frekvens med. Sen har vi också familjen universalinstrument. Varför de överhuvud taget kallas så har alltid varit en gåta för mig. De befinner sig nämligen långt ifrån det 'universella". Vilket av dessa instrument går det t ex att mäta frekvens och kapacitans med?

Nej, vi stackare med 'dammiga' plåböcker är alltför ofta hänvisade till nödlösningar, tur och fantasi. Några extravaganser kan det inte bli tal om för vår del.

HUR KONTROLLERA FREKVENS?

För någon vecka sedan satt jag och funderade över hur i all världen jag skulle bära mig åt för att kontrollera frekvensen på en oscillator, som jag hade byggt, tills det plötsligt slog mig att jag hade en dator. Nåja, det visste jag ju innan också. Däremot hade inte tanken på att utnyttja den som instrument för elektronikmätningar dykt upp tidigare. Men nu gjorde den det.

Plötsligt framstod alla små problem i ett förklarande ljus. Min VIC 64a skulle ju kunna bli ett instrument, som inte skulle gå av för hackor, istället för en anordning att köra mer eller mindre meningslösa program på.

Det räckte med en kontakt till USER-porten och två tåtar, en till jord och en till CNT1-ingången för att jag hårdvarumässigt skulle ha en frekvensmeter (om än en primitiv sådan), som var i stånd att mäta frekvenser upp till flera kHz.

För alla er som inte riktigt fått kläm på hur en frekvensmeter fungerar skall jag försöka förklara principen. titta på blockshemat. (Fig 1.)

Som ni ser har jag döpt blocken till A,B,C,D och E för att lättare kunna prata om dem. Låt oss börja med block A.

Block A: Detta block innehåller i sin primitivaste form en s k amplitudbegränsare. Dess uppgift är att begränsa amplituden på den inkommande signalen till t ex 5V. I klartext innebär det att den utgående signalens storlek blir konstant och tämligen oberoende av den inkommandes storlek.

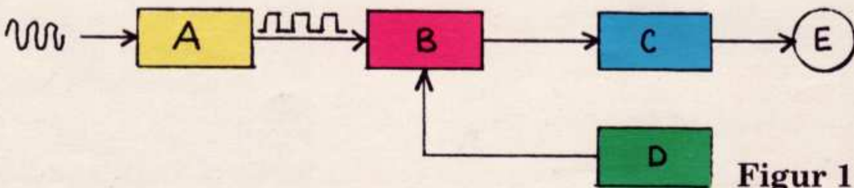
Block B: Detta steg innehåller en elektroniskt styrd strömställare. Under en viss tid, t ex 1 sek, ligger denna strömställare sluten och släpper igenom pulser till block C.

Block C: Detta block håller reda på hur många pulser block B släppt igenom. Har block B släppt igenom t ex 100 pulser under den sekund 'strömställaren' varit sluten är den inkommande signalfrekvensen 100 Hz. Har 1 000 pulser passerat under denna sekund är frekvensen 1 000 Hz, o s v.

Block D: Detta block innehåller själva tidbasenheten, vilken utgöres av en oscillator som har förmågan att ge ifrån sig pulser av noggrannt specificerade tider. Det är detta block som bestämmer hur lång tid vår kära strömställare skall ligga sluten och tillåta signalpulser att passera till block C.

Block E: Detta block består av en anordning som kan avläsa hur många pulser som hamnat i block C och sen presentera resultatet på en sifferdisplay eller, som i vårt fall, på en TV-skärm.

Förloppet är kanske ännu inte självklart. Icke desto mindre ligger det till grund för allt vad den



Figur 1

här artikeln handlar om. Det spelar således ingen roll vad för slags instrument man tänker utnyttja datorn till. Principen går igen i samtliga fall. Sen gäller det bara att hitta den rätta metoden att omvandla s k analoga spänningar och strömmar till frekvenser. Resten sköter datorn om. (Under förutsättning att programmet är det rätta.)

JUMBOJET OCH TRAKTOR

Jag hade nu alltså en frekvensmeter, om än så länge en primitiv sådan. Detta var kanske inte riktigt sant ännu. Vad som i datorhänseende skiljer en jumbojet från en traktor är programvaran. Jag gjorde således ett kort program i BASIC. 825 stod det skrivet på bildskärmen. Saken var nu klar. Oscillatoruslingen gav ifrån sig en frekvens på 825 Hz eller näst intill.

Eftersom man är rädd om sitt kära instrument lödde jag raskt ihop en amplitudbegränsare bestående av två trasistorer. Strömförsörjningen tog jag från USER-porten. Nu skulle inga överspänningar kunna knäcka datorn. Amplitudbegränsaren var dessutom så konstruerad att den inkommande vågformen fick antaga vilka former som helst. Utsignalen förblev en fyrkantvåg.

På USER-porten finns en ingång som heter CNT1. Denna ingång står i förbindelse med en pulsräknare, som i litteraturen kallas för TIMER A.

Denna arbetar i stort sett så att räknaren hämtar ett utgångsvärde från registerna 56580 resp 56581. Tillsammans bildar dessa båda register ett 16 bitars register, och gör det därför möjligt att räkna 65535 pulser. Själva räkningen tillgår så att räknaren läser in innehållet i de båda registren. En puls på CNT1-ingången kommer då att minska dess innehåll med 1. Detta fortgår ända tills alla bitarna räknats ner, då förloppet startar på nytt.

För att räknaren överhuvud taget skall fungera efter våra intentioner måste vi se till att det ligger rätt värde i dess kontrollregister. CRA, som det förkortas, har sin hemvist på adress 56590 (DD0E). Tre bitar måste påverkas här:

BIT 0 En 1a i denna cell startar nedräkningen och en 0a stoppar den.

BIT 4 En 1a här laddar in värdena i registerna 56580 och 56581 till räknaren.

BIT 5 En 1a här kopplar in CNT1-ingången.

Då man matar in ett antal pulser till CNT1 kan man avläsa resultatet i registerna 56580 och 56581. Dessa visar då hur många pulser som återstår från det att inläsningen gjordes. Skulle resultatet bli t ex 45 resp 200 innebär det att det finns 45 + 200 * 256 = 51245 pulser kvar. Således har det inkommit 65535 − 51245 = 14290 pulser. Om detta skett på en sekund sägs frekvensen vara 14290 Hz. Har nedräkningen skett på t ex 10 sekunder blir frekvensen en tiondel, d v s 1429 Hz o s v. Följande förhållande mellan frekvens, pulsantal och tid kan tecknas: f=PA/t, där f=frekvens, PA=antalet inkommande pulser, och t står för den tid under vilken pulserna släpptes in. Vi tar ett exempel till:
Antag att de båda registren 56580 och 56581 avlästes och resultatet blev 200 resp 210. Avsläsningen skedde efter 4 sekunder.
Sök frekvensen.

Återstående pulser:
P=200+210×256=53960
PA:
65535−53960=11517
Frekvensen:
f=11517/4=2893.75 Hz

I dessa exempel förutsatte vi att det totala innehållet i de båda registren var 65535, d v s 255 i vardera registret. Givetvis kan man använda andra utgångsvärden istället.

Som tidbasklocka användes den s k 'JIFFY'klockan vilken indikerar intervaller om 1/60 sek. Denna klocka kan man ställa genom att 'POKEa' in ett värde i register 162. Lägger man in t ex värdet 0 där, behöver man bara vänta tills registerinnehållet blir 60. Tiden som då förflutit är således 60×1/60=1 sek.

Det börjar nu bli dags att visa ett s k flödesschema över vad som händer så att man får en klar överblick. Sånt underlättar och sparar in många magnecyl. Titta nu på fig. 2.

Detta schema visar således bara principen. Förmodligen är det så pass enkelt att det inte tarvar några ytterligare kommentarer.

KAPACITANSMETER

När vi nu kommit så långt att vi fått en fungerande frekvensmeter börjar det bli dags att fundera över vad man kan använda den till mer än mäta frekvenser med. Nämn något. Säg t ex kapacitansmeter, d v s en anordning för att mäta kapaciteten hos kondensatorer. En sån är utmärkt att ha. Vi elektronikknuttar brukar ju ibland ligga inne med ett gäng kondensatorer av okända data.

Vi bygger nu helt enkelt en s k RC-oscillator, som ger ifrån sig en fyrkantvåg med en viss frek-

Forts på sidan 12

Nyheter

Wersiboard 64 – ett nytt keyboard till VIC 64. Wersiboard 64 lanseras av Wersi Electronic. Produkten består av två delar keyboard och programvara.

KEYBOARD

Keyboarden har samma storlek som en professionell keyboard. Den klarar av 49 toner med en intervall på fyra oktaver från C till C. Manualen är idag skriven på 4 språk, dock ej svenska. Man kopplar ihop den till VIC 64 med hjälp av ett intervall.

PROGRAMVARA

Programvaran finns tillgänglig på kassett eller diskett. Polo 64 är en flerstämmig syntheziser. Detta innebär att flera ljud kan ljuda samtidigt t ex flöjter, trumpeter etc. Då produkten ännu är ny har VIC rapport inte fått reda på vem som kommer att distribuera produkten i Sverige. Vi vet därför inte heller priset på produkten. Men är du intresserad av den kan du vända dig direkt till Wersi Electronic, Exportdivision, D 5401, Halsenback för ytterligare information. Man får dock inte köpa mindre än 10 enheter från dem. □

Lätt att använda sprites med COMAL

COMAL är ett mångsidigt "strukturerat" programmeringsspråk. Det utvecklades i Danmark för CBM-datorer och har nyligen kommit fram i s k plug-in-model för VIC 64. Där utnyttjas grafikmöjligheterna och ljudmöjligheterna med VIC 64 helt och vi skall fortsätta den i VIC rapport startade beskrivningen av COMAL-grafiken på VIC 64. Det här är del 3 i den här serien.

Det avancerade men ändå mycket lättanvända programmeringsspråket COMAL finns i flera former för VIC 64. Dels i version 0.12, som är en diskett-laddad version, där man får ca 10Kbytes över för användarens program. Dels finns den nu, när detta läses, levererad till Sverige i plug-in-modul med version 2.0. Den ger ca 30 Kbytes för användarprogram och tar alltså nästan ingenting av det tillgängliga minnet i en VIC 64. Det finns också en version 0.14, som har direktimporterats från USA.

I förra artikeln gavs en översiktlig presentation av den s k Turtle-grafiken, eller "padd-grafiken" som jag kallade den i brist på bättre namn på "The Turtle". Det grafiksystem som ingår i COMAL tar väl vara på de många fantastiska möjligheterna med VIC 64. I COMAL finns också särskilda nyckelord inlagda, med vilka man kan styra s k sprites (användardefinierade figurer). Denna artikel skall ge en snabb översikt över hur lätt det är att använda sprites med COMAL.

På VIC 64 finns, som de flesta 64-ägare väl redan vet, möjlighet att definiera 8 olika figurer, s k sprites. I COMAL används sprite nr 7 för Turtle-figuren. Det lämnar sprites nr 0–6 fria för användning i COMAL. Om man inte använder "paddan" kan man också utnyttja sprite nr 7. Man har i COMAL samma möjligheter som i BASIC att flytta, förstora, vrida och kollidera med sprites. Det är bara så att i BASIC måste alla kommandon ske via PEEK- och POKE-satser medan du i COMAL har en helt speciell uppsättning nyckelord till förfogande.

I föregående artikel fanns en lista över de olika grafik-nyckelorden samt en jämförelse mellan Turtle-grafiken i LOGO och COMAL. Gå tillbaka till den, om något i denna artikel inte är utförligt förklarat. Här finner du en lista över de speciella sprite-kommandon, som ingår i COMAL samt uppgifter om vilka skillnader det är mellan versionerna 0.12/0.14 och 2.0.

VAD ÄR EN SPRITE?

En sprite-figur består av 64 bytes. Dessa 64 bytes används på samma sätt som i BASIC. De kan lagras som en sträng i en DATA-sats, de kan läsas från en diskett-fil eller sättas samman i ett program. När du väl har gjort din 64-bytes-figur låter COMAL dig göra nästan precis vad du vill med den.

Vi börjar med två enkla sprite-figurer för att visa tekniken. Först lägger vi upp två strängvariabler och kallar dem för BOX1$ och BOX2$. Den första figuren är en fylld fyrkant och den andra en randig fyrkant. COMAL låter dig manipulera sprites både i direktmod och inom ett program. Därför kan du antingen ge följande kommandon direkt eller lägga in dem i numrerade programrader.

Vi börjar med att ge kommandot AUTO, så att vi får automatisk radnumrering. Därefter skriver vi in följande program, där vi gör i ordning innehållet i de 64 bytes, som bildar varje "låda", d v s BOX1$ och BOX2$. Vi använder en temporär variabel TEMP och räknare i FOR-slingan och observera att vi bara lägger upp 63 bytes med det tecken vi vill ha i fyrkanten. Den sista byten, dvs den 64e, skall i vårt fall bestå av en nolla, d v s CHR$(0) eftersom vi skall använda högupplösningsgrafik i exemplet.

LÄR DIG PROGRAMMERA

COMAL-GRAFIK PÅ VIC 64



```
0010 SETGRAPHIC 0
0020 DIM BOX1¤ OF 64, BOX2¤ OF 64
0030 FOR TEMP:=1 TO 63 DO
0040  BOX1¤:=BOX1¤+CHR¤(255)
0050  BOX2¤:=BOX2¤+CHR¤(15)
0060 ENDFOR TEMP
0070 BOX1¤:=BOX1¤+CHR¤(0)
0080 BOX2¤:=BOX2¤+CHR¤(0)
     ¤ = $
```



48 OLIKA FIGURER

När du lägger upp en spritebild i COMAL används kommandot DEFINE (=definiera). Därefter anger man definitionsnummer och vilken 64-teckensträng som innehåller denna bild. I COMAL kan du ha upp till 48 olika figurer, numrerade 0–47. Med hjälp av dessa figurer kan du sedan bestämma en av de 8 (eller 7 om "paddan" skall ligga kvar) spritefigurer som du kan ha på skärmen samtidigt. Samma teckning (64-teckensträng) kan användas för flera än en sprite-figur.

```
0090 DEFINE 0,BOX1¤
0100 DEFINE 1,BOX2¤
```

Nu har vi definierat bild 0 och 1 att motsvara de två textsträngar vi hade. Observera att sista bytens nolla är en kontrollbyte, som används av COMAL. Men kom också ihåg nu att denna definitions-sats med DEFINE bara definierar kopplingen mellan sträng och bildnr. Om vi vill ta fram en sprite-figur måste vi skapa en koppling mellan DEFINE-numret och ett sprite-nr. Det sker med kommandot IDENTIFY (= identifiera, känna igen). Vill vi nu identifiera sprite-bild nr 0 och 1 med bilddefinition nr 0 behöver vi skriva ytterligare några programrader:

```
0110 IDENTIFY 0,0
0120 IDENTIFY 1,0
```

Vill vi sedan plocka fram sprites nr 2 och 3 med bilddefinition nr 1 får vi ytterligare några rader:

```
0130 IDENTIFY 2,1
0140 IDENTIFY 3,1
```

Nu vill vi sätta färg på spritefigurerna. Vi vill ha sprite 0 i vit färg och placerad nära mitten på bildskärmen. Då använder vi kommandona SPRITECOLOR och SPRITEPOS:

```
0150 SPRITECOLOR 0,1
0160 SPRITEPOS 0,160,100
0170 FOR X:=1 TO 30 DO SPRITEPOS 1,X,100
```

Sprite-figur nr 1 vill vi ha gul, den skall röra sig tvärs över bildskärmen:

```
0180 SPRITECOLOR 1,7
0190 FOR Y:=1 TO 180 DO SPRITEPOS 2,160,Y
```

Vi vill nu förstora den första sprite-figuren och göra sprite-figur nr 2 i grå färg samt låta den röra sig uppåt på skärmen. Vi vill också att den skall vara hög och smal, d v s förstorad bara i höjd:

```
0200 SPRITESIZE 0,TRUE,TRUE
0210 SPRITESIZE 2,TRUE,FALSE
0220 SPRITECOLOR 2,12
```

TRUE OCH FALSE

Man lägger märke till att båda parametrarna efter SPRITESIZE inte är TRUE nu, utan att en är FALSE. Naturligtvis kan TRUE och FALSE ersättas med sina numeriska motsvarigheter 1 och 0, men om man hellre vill ha värdet 3 för TRUE, sant, så går det lika bra eftersom COMAL anser alla värden som inte är 0 för TRUE. 0 innebär i tillämpligt sammanhang alltid FALSE, falskt. Här betyder det att sprite-bilden endast förstoras i lodrät riktning men inte i vågrät. Rörelsen över skärmen utförs av en FOR-slinga, där SPRITEPOS-talen bestäms gående från 1 till 180, d v s sprite nr 2 behåller läget 160 på x-axeln medan y-axelns tal ändras.

Vi kan slutligen göra sprite-figur nr 3 så att den byter färg medan den rör sig diagonalt över bildskärmen. För nöjes skull gör vi den till en låg, bred figur, d v s vi förstorar den vågrätt.

```
0230 SPRITESIZE 3,FALSE,TRUE
0240 FOR LOCATION:=1 TO 180 DO
0250  SPRITECOLOR 3,RND(0,15)
0260  SPRITEPOS 3,LOCATION,LOCATION
0270 ENDFOR LOCATION
```

Forts på sidan 13

Forts från sidan 9

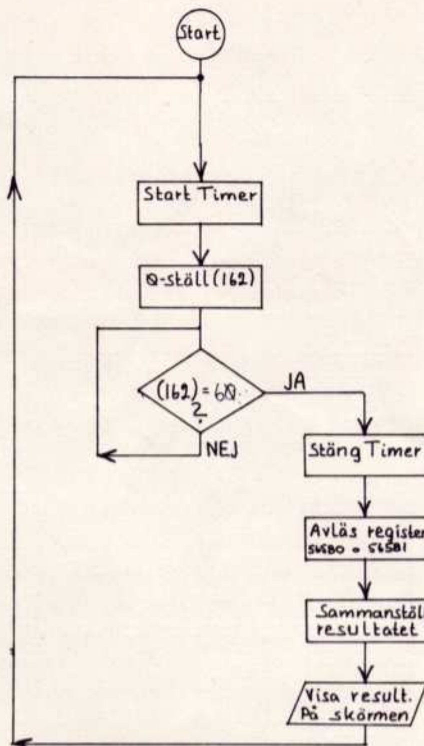
"Utnyttja din 64a..."

vens. (T ex några hundra Hz). Att det heter RC-oscillator beror på att dess frekvensbestämmande element består av motstånd och kondensatorer. Häri ligger lösningen. Motstånd och kondensatorfrekvensen är alltså en f(R,C) om de frekvensbestämmande komponenterna varieras kommer alltså frekvensen att variera. Låter vi motståndsdelen vara fast, bestäms frekvensvariationen endast av kondensatorn och kommer att följa sambandet:
f=K/C
F står för frekvensen, C för kapacitansen och K en konstant som man på laborativ väg kommer fram till när allt är klappat och klart.

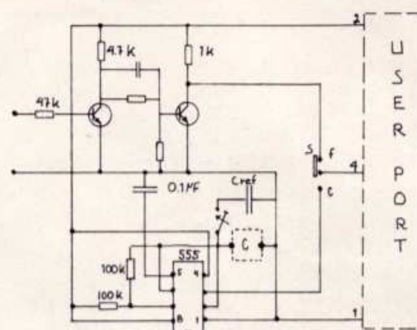
En lämplig oscillator byggs kring den välbekanta 555an, förslagsvis enligt bifogat schema. Kalibreringen tillgår så att man helt enkelt använder en kondensator av känt värde som referens. K i ovanstående formel 'filas' till tills man får ett bra resultat. De båda funktionerna, frekvens och kapacitans, bildar sen ett nytt och användbart instrument, nämligen FC-metern, som ni nog kommer att ha en hel del glädje av i ert idoga pulande. Men låt det bara inte stoppa därmed. Så småningom kommer ni att kunna göra instrument som är i klass med vad som finns i välutrustade laboratorier, men till bara någon promille i kostnad av vad 'riktiga' instrument skulle kosta.

Det publicerade programmet över min FC-meter kan och ska givetvis endast betraktas som ett förslag. Datasatserna är ett litet maskinspråksprogram för att klara den tidskritiska delen, nämligen själva tidbasenheten. Det övriga är mest för att åstadkomma en någorlunda 'stajlföring', så grubbla inte allt för mycket på den. Ni kan säkert komma på något ännu bättre. Till sist ett varningens ord. VICen är en utomordentlig dator. Var rädd om den. Det är nämligen mycket lätt att förstöra kretsar i den. Det behövs bara att man är oförsiktig nog att lägga in en för stor spänning på USER-porten. Det kostade mig 174 kr för en tid sedan plus tre veckors väntetid på en ny krets. □

BN



Figur 2



Figur 3

FC-meterns hårdvara

Cref är en kalibreringskondensator av känd storlek. Med dess hjälp kan K i formeln f=K/C hyfsas till så att korrekt värde på mätningarna erhålls. Själv använder jag en 2.2 µF kondensator och en liten omkopplare för kontinuerlig kontroll av instrumentet. Hela apparaten kan byggas in i en liten plastlåda.

```
0 PRINTCHR$(147)
1 REM ********************************
2 REM *** INLAESNING AV DATASATSER ***
3 REM ********************************
10 FOR X=49152TO49174:READA:POKE X,A:NEXT
15 REM ********************************
16 REM ***         RUBRIK           ***
17 REM ********************************
20 POKE781,2:POKE782,10:SYS65520:PRINT"■                "
30 POKE781,3:POKE782,10:SYS65520:PRINT"■*** FC-METER ***"
40 POKE781,4:POKE782,10:SYS65520:PRINT"■                "
45 REM ********************************
46 REM ***       ALTERNATIV         ***
47 REM ********************************
50 POKE781,12:POKE782,1:SYS65520:PRINT"■1=FREKVENS  "
60 POKE781,13:POKE782,1:SYS65520:PRINT"■2=KAPACITANS"
70 GETA$:IF A$="1"THENU=0:GOTO105
80 IF A$="2"THENU=1:GOTO105
90 GOTO 70
105 REM ********************************
106 REM ***      TILL TIDBASEN       ***
107 REM ********************************
110 SYS49152
120 REM ********************************
121 REM *** AVLAESNING AV TIMER A    ***
122 REM ********************************
150 A=PEEK(56580):B=PEEK(56581)
160 P=65535-(A+256*B)
165 REM ********************************
166 REM ***   FREKVENS-KAPACITANS    ***
167 REM ***   FUNKTIONSRUBRIKER      ***
169 REM ********************************
170 IF P=0THENC=0:GOTO200
180 IFU=0THENC=P
190 IFU=1THENC=INT(8800/P)
200 IFU=0THEN POKE781,5 :POKE782,10:SYS65520:PRINT "■ * FREKVENS *  "
205 IFU=1THEN POKE781,5 :POKE782,10:SYS65520:PRINT "■ * KAPACITANS * "
210 IFU=0THEN P$="■F ="+STR$(INT(C))+" HZ     "
220 IFU=1THEN P$="■C ="+STR$(INT(C))+" NF     "
225 REM ********************************
226 REM *** PRESENTATION AV RESULTAT***
227 REM ********************************
230 POKE781,20:POKE782,20:SYS65520:PRINT P$
240 GETA$:IFA$="1"THENU=0
250 IFA$="2"THENU=1
260 GOTO105
996 END
997 REM ********************************
998 REM ***       'MASKIN'KOD        ***
999 REM ********************************
1000 DATA 169,49,141,14,221,169,0,133,162,169,60,234,197,162,208,252,169,33,141
1010 DATA 14,221,24,96
READY.
```

Forts från sid 11

"Lätt att använda sprites..."

Här stannar vi och avslutar AUTO-numreringen med en extra tryckning på RETURN utan att skriva något, för 2.0-COMAL anv STOP-tangenten. För att se hur vårt program fungerar skriver vi RUN och trycker RETURN. Vill du se programmet igen, skriv RUN och tryck RETURN igen. Vill du att programmet skall upprepa sig självt, tryck på STOP och skriv LIST samt lägg in följande programrader:

165 REPEAT
280 UNTIL KEY$<>CHR$(0)

Nu fortsätter programmet ända till dess du trycker på någon tangent.

ATT DEFINIERA EN SPRITE I COMAL PÅ ETT ANNAT SÄTT

Det finns ett särskilt program i COMAL som underlättar spritedefinitioner. För att definiera en sprite i COMAL 0.12 måste först programmet SPRITEDESIGN laddas in. Det sker med kommandot:

CHAIN "SPRITEDESIGN" <RETURN>

Nu börjar diskettstationens lampa att lysa och om du har COMAL-disketten inne börjar programmet att laddas in. Du ser på skärmen att en fyrkant märks ut på vänster sida av bildskärmen samt att det kommer upp ett antal frågor i övre högra hörnet. Den första:

MULTICOLOR?

skall besvaras med 0 eller 1. Svarar du 0 kommer sprite-figuren att ritas i s k högupplösningsgrafik, d v s med 24*21 små fält, som kallas pixels (Picture elements, bildelement). Svarar du 1 kommer du att ha möjlighet att göra en flerfärgs sprite-figur, men inte med lika fina detaljer som med högupplösningsgrafik.

Vi antar att vi svarar 0. Då skriver programmet följande fråga:

COLOR:

därefter väntar det på att vi skall ange ett färgnummer (se tabell) Vi kan t ex slå in <CTRL 8>, som betyder gul. Sedan får vi en fråga om vi vill öka ut figuren i sidled (vågrätt):

EXPANDX?

Svaret kan vara 0 eller 1. Skriver man 0 så ritas sprite-figuren i normal storlek i vågrät riktning. Skriver man 1 så blir figuren dubbelt så stor vågrätt, d v s man EXPANDerar figuren i X-axelns riktning. Låt oss ange 1 för att få en mycket tydlig figur på skärmen. Härefter skriver programmet:

BACKCOL:

och det är vår uppgift att ange bakgrundsfärg. Det sker med samma färgkoder som tidigare, se tabellen. Skriv in t ex C= och 4 så får du en grå bakgrund.

I och med att alla frågor nu besvarats kvitterar programmet med att rita upp en mindre kvadrat under "frågeschemat". Markören placeras i övre vänstra hörnet på den stora kvadraten. Den lilla kvadraten visar hur stor figuren egentligen blir, den stora är till för att man enkelt skall kunna definiera sin sprite-figur.



MARKERING FÖR SPRITE-FIGUR

I samtliga fält, kom ihåg att det i högupplösningsgrafik finns 24*21 fält, skall nu anges om fältet skall synas i förgrundsfärgen (svaret efter COLOR) eller i bakgrundsfärg (svaret efter BACKCOL). Denna markering sker med siffertangenterna 0 och 1. Man ger färg åt det fält där markören står genom att trycka på 0 om det skall ha COLOR-färgen och på 1 om det skall ha BACKCOL-färg. (Nu används således inte färgkoderna från tabellen längre, de används endast för att definiera färgerna vid COLOR och BACKCOL.)

Man kan när som helst ändra en inställd färg genom att flytta markören med de vanliga markörflyttningstangenterna och ange om man vill ändra färgkod från 0 till 1 eller tvärtom.

Under färgmarkeringen kommer du att märka att den lilla kvadraten fylls i allt eftersom du knappar in 0or och 1or. Där ser du alltså din sprite-figur i den storlek som den kommer att få när du vill använda den.

Om du nu vill spara din figur på diskett för senare framtagning och användning i ett program trycker du på tangenten med bokstaven "s" för SAVE. Då skriver sprite-editorn på skärmen:

FILENAME:

och du skriver in ett godkänt filnamn och trycker på RETURN. (Du hade väl en skiva i skivminnet?) Hämtning sker med kommandot OPEN FILE och READ FILE.

FLERFÄRGADE SPRITE-FIGURER

Om vi i början av frågeserien hade svarat med 1 på frågan MULTICOLOR?, d v s valt flerfärgsgrafiken, hade vi fått två följdfrågor som kommit upp på skärmen:

MULTICOLOR−1:
MULTICOLOR−2:

För var och en av dessa skall man ange färgkoder. Dessa två färger kallas också för sidofärger. Vid flerfärgs-grafik får man bestämma färgerna i fält, som i vågrät riktning är dubbelt så stora som i högupplösningsgrafik, d v s de består egentligen av två bredvid varandra liggande fält från högupplösningsgrafikens ruta. Fördelen är att man nu har fyra färger att tillgå, nackdelen att figuren blir mindre detaljrik. Färgkoderna hämtas från kodtabellen, där 0=svart och 15=grå.

Dessa två färger kallar vi sidofärger. Denna gång skall vi alltså använda totalt fyra färger, en bakgrundsfärg, en förgrundsfärg och två sidofärger. Nu behöver man tydligen använda fyra tangenter för att på varje markörläge ange vilken färg man vill ha. Javisst, vi använder tangenterna 0, 1, 2 och 3! 0=bakgrundsfärg, 1=förgrundsfärg och 2=sidofärg 1 och 3=sidofärg 2.

Forts på sidan 14

Har du ett bra recept
på någon maträtt?
Dela med dig, och ring till
VIC rapports databas
Tel 08-19 06 16
Kodord 90000VIC64

Bli prenumerant på
VIC rapport!
Pg 846 46–9
120:–/år

Forts från sidan 13

LÄR DIG PROGRAMMERA

ÄNDRING AV SPARAD SPRITE-FIGUR

Vill man av någon anledning ändra en redan ritad och sparad figur är det lätt gjort med COMAL. Man startar upp rutinen SPRITEDESIGN igen och när markören står i ritkvadratens övre vänstra hörn trycker man på tangenten med bokstaven "l" för LOAD och får frågan:
FILENAME?
Där skriver man i namnet på den sekvensiella fil där den sprite man vill ändra är sparad. Sedan, när programmet ritat upp figuren kan man använda markörtangenterna som vanligt för att flytta markören och ändra olika fält som man vill. Då ändringen är klar kan man som vanligt spara den nya genom att ge den ett nytt namn. Vill man radera den gamla så används kommandot DELETE filnamn.

(Se färgtabell nedan)

KOLLISIONER, KROCKAR OCH "OMBORDLÄGGNINGAR"

I COMAL kan du också ange om de data som finns på skärmen har FÖRETRÄDE framför andra data genom nyckelordet PRIORITY, som betyder just företräde, prioritet. Kollisioner mellan sprites och data rapporteras av en särskild funktion som i COMAL heter DATACOLLISION-funktionen. Kollisioner mellan sprites överförs genom en annan funktion: SPRITECOLLISION-funktionen.

Dessutom kan du "gömma" sprites genom kommandot HIDESPRITE (=göm sprite). Med kommandot SHOWSPRITE får du fram din sprite-figur igen. Titta i tabellen över nyckelord och syntax så att du gör rätt, den var införd i föregående nummer av VIC rapport.

Du, som har skaffat dig COMAL-disketten 0.12 från din VIC-handlare, kan pröva att använda den andra av ovan nämnda metoder för sprite-figur-ritning, SPRITEDESIGN-programmet, som också finns på disketten. Sedan kan du pröva att använda din nya sprite-figur i t ex programmet "upp-mot-tak" eller det modifierade "STUDSANDE FIGURER" som vi återger här nedan.

Åke Fredriksson

Färgtabell i COMAL

0 SVART	1 VIT	2 RÖD
3 CYANBLÅ	4 PURPUR	5 GRÖN
6 BLÅ	7 GUL	8 ORANGE
9 BRUN	10 LJUSRÖD	11 GRÅ 1
12 GRÅ 2	13 LJUSGRÖN	14 LJUSBLÅ
15 GRÅ 3		

```
0005 // SAVE "STUDSFIGUR" AKTUELL VERSION
0010 DIM FIGUR¤ OF 64
0020 TRAP ESC-
0030 TASPRITE
0040 WHILE TRUE DO
0050  TECKNASPRITE
0060  DRATAK
0070  ANVSPRITE
0080 ENDWHILE
0090 //
0100 PROC TASPRITE
0110  OPEN FILE 2,"SPRITEKLADD",READ
0120  READ FILE 2: FIGUR¤
0130  CLOSE
0140 ENDPROC TASPRITE
0150 //
0160 PROC TECKNASPRITE
0170  DEFINE 45,FIGUR¤
0180  SPRITESIZE 2,1,1
0190  BACKGROUND 8
0200  BORDER 2
0210  SPRITECOLOR 2,11
0220  SETGRAPHIC 0
0230  IDENTIFY 2,45
0240 ENDPROC TECKNASPRITE
0250 //
0260 PROC ANVSPRITE
0270  ITAKET:=FALSE
0280  FOR X:=0 TO 319 DO
0290   Y:=2/3*X
0300   SPRITEPOS 2,X,Y
0310   X1:=X
0320   ITAKET:=DATACOLLISION(2,TRUE)
0330   IF ITAKET THEN NER'IGEN
0340  ENDFOR X
0350 ENDPROC ANVSPRITE
0360 //
0370 PROC STOPPASPRITE
0380  PENCOLOR 8
0390  BORDER 8
0400  BACKGROUND 11
0410  SETTEXT
0420  TRAP ESC+
0430  PRINT CHR¤(14),CHR¤(147)
0440  STOP
0450 ENDPROC STOPPASPRITE
0460 //
0470 PROC DRATAK
0480  PENCOLOR 11
0490  MOVETO 0,180
0500  DRAWTO 319,180
0510 ENDPROC DRATAK
0520 //
0530 PROC NER'IGEN
0540  FOR X:=X1 TO 0 STEP -1 DO
0550   Y:=2/3*X
0560   SPRITEPOS 2,X,Y
0570  ENDFOR X
0580  IF ESC THEN STOPPASPRITE
0590 ENDPROC NER'IGEN
0600 //
0700 // OBSERVERA ATT DU SKALL LAGRA UNDAN DIN SPRITEFIGUR MED NAMNET
0710 //"SPRITEKLADD" ELLER OCKSA MASTE DU ÄNDRA I PROGRAMMET SA ATT DITT FIGUR-
0720 //NAMN STÄMMER MED PROGRAMMETS!!!!
0730 // OBSERVERA ATT DETTA ÄR VIKTIGT!
```

program studsande figur

Tävlingsresultat

Så var det dags att presentera resultaten av den tävling som utlystes i VIC rapport nr 5/6.

Redaktionen har fullkomligt översvämmats av bidrag och vi har haft många roliga timmar när vi har gått igenom alla bidrag. P g a det stora tävlingsdeltagandet har vi inte ens möjlighet att publicera alla de bra program som vi har fått ta del av. För att underlätta för oss själva har vi beslutat att utlysa inte mindre än 4 förstapris, 4 andrapris och 4 tredjepris samt en del hedersomnämnanden.

De fyra vinnarna är uppdelade enligt klasserna: Spelprogram, Nyttoprogram, VIC 64 och VIC 20. Vi har inte möjlighet att publicera alla vinnarna i detta nummer och för att öka spänningen har vi beslutat att presentera de fyra tredjepristagarna i detta nummer, andrapristagarna i nr 10 och förstapristagarna i nr 11/12.

Nedan kommer de fyra tredjepristagarna att publiceras tillsammans med juryns omdöme. Det kan i sammanhanget nämnas att juryn består av följande fem personer: Ola Axelsson, Joakim Aspengren, Matts Nilsson, Elisabeth Höglund och Nina Linander. *Nu till presentationen av tävlingen.*

PANIC

```
0 REM        --------------------------
1 REM        PANIC AV ANDERS ERIKSSON
2 REM        ------- 84 07 21 -------
3 REM        --------------------------
10 DIMPO(20):DIMHI(20)
20 PRINTCHR$(147):POKE646,1:POKE53281,0:POKE53280,0
30 S=54272:POKES+23,0:POKES+1,0:POKES+24,15:POKES+5,40:POKES+6,201
40 X=20:Y=10:R=1:Z=Z-1:X1=5:Y1=5:SC=-1
50 FORI=1TO19:PO(I)=1500:NEXTI
60 PRINT"
70 PRINT"                                            "
80 PRINT"                                            "
90 PRINT"                                            "
100 PRINT"                                           "
110 PRINT"                                           "
120 PRINT"
130 PRINT"           AV ANDERS ERIKSSON (C)"
140 PRINT"           - MOTSOLS          - MEDSOLS"
150 INPUT"           HUR LA[…]NG MASK (1-18) ";Z:IFZ<1ORZ>18THEN150
160 Z=Z+1:PRINT"VA[…]NTA, SKA[…]RMEN GO[…]RS IORDNING."
170 FORI=1024TO1984STEP40:POKEI,160:POKEI+54272,5:NEXTI
180 FORI=1063TO2023STEP40:POKEI,160:POKEI+54272,5:NEXTI
190 PRINT"HISCORE:"HI(Z-1)
200 FORI=55296TO56295:POKEI,1:NEXTI
210 FORI=1024TO1103:POKEI,160:POKEI+54272,5:NEXTI
220 FORI=1944TO2023:POKEI,160:POKEI+54272,5:NEXTI
230 PRINT"HISCORE:"HI(Z-1)
240 PRINT"MASKLANGD:"Z-1
250 GETS$:IFS$<>""THEN250
260 GOTO400
270 GETA$:IFA$<>""THEN370
280 IFR=1THENX=X+1:GOTO320
290 IFR=2THENY=Y+1:GOTO320
300 IFR=3THENX=X-1:GOTO320
310 IFR=4THENY=Y-1
320 PO(Z)=1024+X+40*Y
330 IFPEEK(PO(Z))<>32THEN400
340 POKEPO(Z),81:POKEPO(1),32
350 FORI=1TOZ-1:PO(I)=PO(I+1):NEXTI
360 GOTO270
370 IFA$=".""THENR=R+1:IFR=5THENR=1
380 IFA$=","THENR=R-1:IFR=0THENR=4
390 GOTO280
400 IFPEEK(PO(Z))<>102ANDPEEK(PO(Z))<>32THEN530
410 POKES+4,33:POKES+1,100:POKES+4,32
420 POKE1024+X1+40*Y1,32 :POKE1024+X1+40*Y1+1,32
430 X1=INT(37*RND(1))+1:IFABS(X1-X)<3THEN430
440 Y1=INT(20*RND(1))+2:IFABS(Y1-Y)<3THEN440
450 IF PEEK(1024+X1+40*Y1)<>32ORPEEK(1025+X1+40*Y1)<>32THEN430
460 X2=INT(37*RND(1))+1:IFABS(X2-X)<3ORX2=X1THEN460
```



VILKA BIDRAG VI HAR FÅTT!

3:e pris Spelprogram VIC 64

***Tävlingsbidrag:* PANIC**
av Anders Eriksson
***Utrustning:* VIC 64 bandspelare/diskettstation**
***Betyg:* 3**

Omdöme: Detta är den välkända 'MASKEN' som finns i ett antal versioner till olika datorer. Det går ut på att styra en 'mask' och samtidigt äta upp så många 'prickar' som möjligt, detta medan masken blir längre och längre. Det slumpas bara ut en 'prick' i taget. Styrningen är det svåraste; två tangenter – en medsols och en motsols.

Du får inte 'köra på' dig själv eller kanterna, i så fall är det 'GAME OVER'.

Nu till denna variant: Istället för att masklängden ökar så uppkommer det en liten fyrkant någonstans på skärmen som man inte får krocka med. Det uppkommer EN varje gång man äter en 'prick'.

I början väljer man alltså masklängd själv (1–18) och ju mindre masken är desto snabbare går den. Här håller också programmet reda på HI-SCORE (högsta poängen) för varje masklängd.

Grafik & ljud: Det är vanlig tangentbordsgrafik och det finns bara två ljud: ett vid "prickätandet" och ett vid krocken för det finns ju inga fler tillfällen att använda ljudgeneratorn.

Forts PANIC

```
470 Y2=INT(20*RND(1))+2:IFABS(Y2-Y)<3ORY2=Y1THEN470
480 POKE1024+X2+40*Y2,79 :POKE1024+X2+40*Y2+1,80
490 POKE1024+X2+40*Y2+40,76 :POKE1024+X2+40*Y2+1+40,122
500 POKE1024+X1+40*Y1,102:POKE1024+X1+40*Y1+1,102
510 SC=SC+1:PRINT"■■     SCORE:"SC
520 GOTO340
530 POKES+4,33:FORT=1TO40:FORE=1TO50STEP5:POKES+1,T+E:NEXTE:NEXTT:POKES+4,32
540 GETS$:IFS$<>""THEN540
550 PRINT"■■■■■■DU FICK"SC"POA■■NG."
560 PRINT"■■■■MASKLA■■NGDEN VAR"Z-1"■."
570 PRINT"■■■■HISCORE VAR FO■■RUT"HI(Z-1)"POA■■NG."
580 IFSC<=HI(Z-1)THEN600
590 HI(Z-1)=SC:PRINT"■■■■DU HAR NYTT HISCORE MED■";SC;"■POA■■NG.":GOTO610
600 PRINT"■■■■REKORDET STA■■.■R SIG..."
610 PRINT"■■          ■ TRYCK PA EN KNAPP ■"
620 GETS$:IFS$=""THEN620
630 GOTO20
READY.
```

FUEL LANDER

```
READY.

0 PRINT"(CLR)":D=1:POKE36878,15:POKE36879,8
1 POKE646,D:D=D+1:PRINT"(HOME)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RVS)  (RGHT) (RGHT)
 (RGHT)  (RGHT) ":POKE36878,10:IFD=8THEND=1
2 PRINT"(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RVS) (RGHT)(RGHT) (RGHT) (RGHT) (RGHT)(RG
HT) ":POKE788,194:REM"(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)(DE
L)(DEL)(DEL)(DEL)(DEL)(DEL)
3 POKE36878,7:PRINT"(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RVS)  (RGHT) (RGHT) (RGHT)  (
RGHT) "
4 PRINT"(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RVS) (RGHT)(RGHT) (RGHT) (RGHT) (RGHT)(RG
HT) "
5 PRINT"(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RVS) (RGHT)(RGHT)(OFF)◥(RVS) (OFF)◤(RVS)(
RGHT)  (RGHT)  "
6 POKE36878,5:POKE646,D:PRINT"(DOWN)(RVS) (RGHT)(RGHT)◤ ◥(RGHT) (RGHT) (RGHT)  ◥
(RGHT)  (RGHT)  ◥"
7 PRINT"(RVS) (RGHT)(RGHT) (RGHT) (RGHT) ◥ (RGHT) (RGHT) (RGHT) (RGHT)(RGHT) (RG
HT) "
8 POKE36878,3:PRINT"(RVS) (RGHT)(RGHT)   (RGHT)   (RGHT) (RGHT) (RGHT)  (RGHT)
(OFF)◤"
9 PRINT"(RVS) (RGHT)(RGHT) (RGHT) (RGHT) (OFF)◥(RVS) (RGHT) (RGHT) (RGHT) (RGHT)
(RGHT)  ◥"
10 POKE36878,2:PRINT"(RVS)  (RGHT) (RGHT) (RGHT) (RGHT) (RGHT)  (OFF)◤(RVS)(RGHT
)  (RGHT) (OFF)◥(RVS) "
15 PRINT"(HOME)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN
)(DOWN)(BLU)(RGHT)(RGHT)USE YOUR JOYSTICK.":POKE775,0:REM"(DEL)(DEL)(DEL)(DEL)(D
EL)(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)
20 POKE36878,1:PRINT"(DOWN)(RGHT)(RGHT)(YEL)TRY TO LAND ON THE     FUELBASE SAFT
LEY     TO GET MORE FUEL."
30 PRINT"(DOWN)(RGHT)(RED)TIP: DO IT CAREFULLY!(DOWN)(RGHT)(RGHT)(RGHT)(RGHT)(RG
HT)(RVS)(CYN)PRESS RETURN!"
35 POKE36878,15:FORA=150+D*13TO200+D*7:POKE36876,A:NEXT:POKE36876,A-5
40 GETA$:IFNOTA$=CHR$(13)THEN1
45 POKE36876,0:READA,B:V=A+B:FORI=ATOA+B:READC:POKEI,C:V=V+C:NEXT
50 PRINT"(CLR)(YEL)*** SCORE RANKING ***":PRINT"(DOWN)(DOWN)(DOWN)(DOWN)(WHT)WHI
TE = 100":PRINT"(DOWN)(RED)RED = 200"
60 PRINT"(DOWN)(CYN)CYAN = 300":PRINT"(DOWN)(PUR)PURPLE = 400":PRINT"(DOWN)(GRN)
GREEN = 500":PRINT"(DOWN)(BLU)BLUE = 600"
70 PRINT"(DOWN)(YEL)YELLOW = 700(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)
(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(DOWN)(DOWN)(RVS)(CYN)PRESS RETURN!"
73 IFPEEK(197)<>15THEN73
75 PRINT"(CLR)(YEL)(DOWN)(DOWN)LOAD FUEL LANDER 2 ,1(HOME)";
80 POKE7728,34:POKE7742,34:NEW:REM"(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)
90 DATA7087,80,162,27,160,175,134,56,132
100 DATA55,162,27,160,194,142,21,3,140,20
110 DATA3,96,169,127,141,34,145,169,0,133
120 DATA0,141,19,145,173,17,145,72,106,106
130 DATA162,7,32,245,27,104,42,42,162,128
140 DATA32,245,27,173,32,145,106,106,106
150 DATA106,162,8,32,245,27,169,255,141,34
160 DATA145,76,191,234,134,252,37,252,69
170 DATA252,5,0,133,0,96

READY.
```

```
READY.

0 REM"(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)**** FUEL  LANDER ****
 BY MATTIAS HOFFBLAD
1 REM"(DEL)(DEL)(DEL)(DEL)(DEL)(DEL)                    A HOME-MADE PRODUCTION
10 PRINT"(CLR)":POKE36879,8:FORA=36874TO36878:POKEA,0:NEXT:FORA=7176TO7415:READB
:POKEA,B
20 NEXT:POKE775,0:FORA=7424TO7431:POKEA,0:NEXT:POKE36869,255:POKE36879,8:CLR
30 PRINT"(HOME)(RVS)(YEL)**** FUEL  LANDER ****(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DO
WN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN
)(DOWN)(OFF)(BLU)EFG(CYN)AAAAAAAAAAAAAA"
35 POKE7798,24:POKE7846,24:POKE7734,24:POKE7859,24:POKE7810,24:POKE7730,24:POKE7
865,24
40 PRINT"(BLU)HIJ(RVS)(CYN)"SC"(UP)(UP)"
42 POKE37879,2:POKE646,RND(1)*7+1:PRINT"(HOME)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOW
N)(DOWN)(DOWN)(DOWN)RTSTSR(RGHT)(RGHT)RS":PRINT"MMMMMV(RGHT)UMMN"
43 PRINT"MMMMV(RGHT)(RGHT)MMMMO"
44 PRINT"MMMV(RGHT)(RGHT)(RGHT)MMMMMN(RGHT)(RGHT)(RGHT)TSRSTS";
45 PRINT"MMV(RGHT)(RGHT)(RGHT)QMMMMMMP(RGHT)UMMMMMM";
46 PRINT"MM(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)WMMMM(RVS)◤(OFF)(RGHT)(RGHT)MMMMMMM";
47 PRINT"MMO(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)WMMV(RGHT)(RGHT)WMMMMMMM";
48 PRINT"MMMO(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)MM(RGHT)(RGHT)(RGHT)(RGHT)WM(RVS)◤(OF
F)(RGHT)MMM";
49 PRINT"MMMMNRS(RGHT)UMMOS(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)MMM";
50 PRINT"MMMMMMMMMMMMMMMMMMMMMM";
51 PRINT"MMMMMMMM(RVS)(YEL)LEVEL= 1";:POKE646,PEEK(38730):PRINT"(OFF)MMMMMMM"
59 DATA170,170,170,170,170,170,170,170,168,168,168,168,168,168,168,168
60 DATA160,160,160,160,160,160,160,160,128,128,128,128,128,128,128,128,,,234,138
,234
65 DATA138,142,,,,232,136,232,136,238,,,,,48,,48,,,,,238,138,232,42,238,
80 DATA,,238,170,174,172,234,,,,224,131,224,131,224.
```

3:e pris Spelprogram VIC 20

Tävlingsbidrag:
FUEL LANDER
av Mattias Hoffblad
***Minne:* Oexpanderad VIC 20**
***Betyg:* 4**

Omdöme: FUEL LANDER – ett spel för dem med nerver – går ut på att förflytta ett rymdskepp långt, långt nere i månens grottor och avgrunder, i vilka det förekommer s k bränslebaser.

Dessa bränslebaser är nödvändiga för att du skall kunna ta dig fram med ditt rymdskepp, men bränslet är oftast svårtillgängligt allra längst ned i de djupaste grottorna, varför du måste hantera din joystick med största uppmärksamhet.

Svårighetsgraden ökar efter hand som du klarar dig till bränslebaserna. Sammantaget finns åtta (!) svårighetsgrader.

Såväl grafik som spelutförande är utmärkt. Spelet är i höguppplösningsgrafik, och spelplanen hör därför till en av tävlingens bättre.

Som jag tidigare nämnde blir spelet svårare och svårare desto längre tiden går, vilket får anses vara bra, eftersom man då hinner "träna" in sig i början för att sedan få svårare utmaningar.

Positivt med spelet är även att det är jämförelsevis kort, och att det inte på minsta sätt har hämmat spelets enastående höguppplösningsgrafik. □

Tävlingsresultaten fortsätter på sidan 30

Programmet fortsätter på sidan 30

Klaviatur till VIC 64

Med anledning av intresset för en klaviatur anpassad till VIC 64, så att man kan använda SID-Chippet för eget musicerande, kan jag glädja läsarna med att detta nu finns färdigt och kommer att börja säljas i höst.

Klaviaturen är 49 fullstora tangenter som standard. Styrprogrammet är i maskinkod och ligger i cartridge. Ur denna cartridge går sedan en flatkabel till klaviaturen. Hela enheten är mycket nätt. Första bilden är en meny över fasta ljud. Efter val av ljud får man en bild av en manöverpanel till en synthesizer med skjutpottar och rattar. Dessa står då i den inställning som motsvarar det ljud man valt. I detta läge kan man spela. Om man inte är nöjd med ljudet kan man justera till önskat sound genom att manövrera rattar och pottar på skärmbilden. Detta görs på ett enkelt och snillrikt sätt med 64ans tangentbord.

Man kan justera effekter som A, D, S, R, vågform, typ av filter, resonansfrekvens, pulsform, chorus, transponering mellan olika tonarter och volym. Andra finesser är att man kan spela in ett musikstycke för att spara på tape eller låta 64an upprepa stycket medan du själv slår dig ner i fåtöljen och försöker upptäcka om du spelat fel någonstans. En annan använding av "Recordmetoden" är att du spelar duo med 64an.

Möjligen kommer klaviaturen att säljas både färdig och som byggsats. Priset på en komplett klaviatur och cartridge torde hamna runt 1 200:–.

Håkan Holmström

Tips, frågor och artiklar! Skriv till VIC rapports redaktion!

Kopiera skärmen med printer 1526 (för VIC 20)

Har man en printer är det ett starkt önskemål att man ska kunna få ut en kopia av det som visas på bildskärmen. Ett aktuellt fall för mig var att få ut kopior på kurvor framtagna med hjälp av super expander-kassetter för högupplösningsgrafik.

Speciellt lämplig för att kopiera dessa borde printern 1526 vara med sina åtta nålar, som perfekt kan avbilda varje tecken och mönster. Det var därför med viss besvikelse som det noterades att manualen ej beskrev någon dylik möjlighet. Detta förhållande har även omnämnts i litteraturen t ex i tidskriften Compute. Enda möjligheten var att definiera ett enda eget specialtecken på endast 8*8 bitar.

Skam den som ger sig. Genom ett kort samtal med Kjell Segerbom på Handic bekräftades att man med lite möda kan kopiera skärmen. Det gäller att utnyttja specialtecknet, som bara kan ändras från rad till rad.

Detta problem löses genom att sätta radframmatningen till noll och på så sätt låta flera rader skriva över varandra. För att inte printern ska behöva slita ihjäl sig måste utskrift även kunna ske baklänges. Detta går fint med 1526 om man går rätt till väga.

Här beskrivs en rutin för kopiering av skärmen oavsett storlek och var den och teckensatsen lagrats i minnet. Detta är ju ett speciellt problem på VIC 20. Med superexpander i grafikmod erhålles en skärm på 20*20 rutor med 160*160 punkters upplösning. Bättre grafik upp till 32*30 rutor är dock möjligt med extra minne och annan disposition av detta. Se t ex vissa spel.

METOD

Grafiken är på VIC 20 lagrad upp till 512 tecken. Normalt används 22*23=506 st. Superexpander använder blott 20*10=200 st. Att superexpander ej behöver 20*20=400 st beror på att varje tecken här definieras som ett 16*8 punkters mönster mot det normala på 8*8 punkter. Ett tecken på 8*8 punkter kräver 8 bytes för lagring i minnet. Totalt kräver således superexpander 3200 bytes för plottning av god-

tycklig grafik. Härtill kommer att även de 200 tecknen måste lagras. Detta är inte möjligt med en oexpanderad VIC 20 och således har man försett superexpander med 3K (3072) bytes extra minne.

Eftersom grafiken lagras i form av tecken så indelas skärmen i ett rutmönster där varje ruta innehåller ett tecken. Varje tecken som lagras i form av bytes har dessa definierade enligt figuren.

Byte 0

Byte 8

Tecken om 8*8 punkter

Som synes lagras informationen radvis med den minst signifikanta biten åt höger. Detta format är dock olämpligt för 1526-printern som med sina åtta nålar skriver ut punkterna kolumnvis och således börjar med den mest signifikanta biten till vänster. En specialrutin behövs således för att definiera om varje tecken så att ett lämpligt format erhålles för printern. Printerns specialtecken definieras och överförs i form av en sträng om 8 bytes representerande de 8 kolumnerna. Den minst signifikanta biten ligger längst ner. Att göra detta i Basic är möjligt men skulle ta lång tid. Därför har en liten maskinkodsrutin konstruerats som direkt definierar strängen i Basic-minnet.

För att kunna ta varje 8*8 cell i rätt ordning undersöks om varje tecken består av 8*8 eller 16*8 punkter (NH). Sedan kontrolleras antalet rader (NR) och kolumner (NC) på skärmen. Naturligtvis måste även skärmminnets och teckenminnets startadresser (SM och CM) vara kända. På så sätt kan minnesläget för varje tecken beräknas. Det lagras sedan på sidan noll i minnescellerna 251 och 252. Härifrån hämtar maskinkodsrutinen sin information och definierar specialtecknet i PR$. Därefter skrivs tecknet ut.

För varje nytt specialtecken ändras dess läge genom strängen PP$ som består av blanktecken. Dessa påverkar dock ej printerhuvudet så att det behöver röra sig tillbaka till radens start, utan det rör sig bara lite ryckigt fram och tillbaka på raden. Utskrift av den visade stjärnfiguren tar drygt en minut. Med en glest fylld skärm går det snabbare eftersom tomma celler ej skrivs ut.

För den som ännu inte har någon printer kan nämnas att specialtecknet skrivs ut genom att använda CHR$(254).

PRINT#6,CHR$(0) eller CHR$(21) sätter radframmatningen till 0 eller 21. Det senare värdet ger lagom täta rader per sida. PRINT#3,CHR$(114) sätter antalet rader per sida till 114 och PRINT#4,CHR$(147) kopplar in automatiskt sidbyte.

MASKINKODSRUTIN

Först något om lagringen av maskinkodsrutinen då detta beredde vissa problem. Med 8K eller mer i expansion kan visserligen det lediga 3K utrymmet vid 1024 utnyttjas ty superexpander använder bara minnet mellan adresserna 4096–8192, men det ska ju fungera även med bara 3K extra minne. Problemet beror på att superexpander använder nästan allt ledigt utrymme som det mellan 673–767 och kassettbufferten 828–1019. Dessutom flyttas pekarna för start och slut på Basicminnet.

Problemet har lösts på ett originellt sätt genom att lagra maskinkoden i form av en sträng (som maximalt kan bestå av 255 bytes). Strängen måste definieras vid programkörningen annars lagras den i programtexten och kan sabotera en ny körning. Det är också väsentligt att koppla in grafikmoden från början så att strängläget ej ändras. Detta påverkar vissa minnespositioner på sid 0 som senare används i maskinkoden. Detta sker på raderna 110 och 120 i Basicprogrammet. Inläsning av maskinkoden sker på raderna 130–210. Maskinkoden ges även i en assemblerlista separat. Låt oss se hur rutinen fungerar, genom att studera assemblerlistan.

Programlistning se sid 35

HITTAR STRÄNGVARIABELN

Adresserna längst till vänster är de som erhålls med 16K expansion och är givna hexadecimaltal d v s med alfabetet (0,1...15). Först lagras byten 50 och D2 i minnescellerna $45 och $46. Dessa byte representerar strängnamnet PR. Bokstaven R representeras i ASCIIkod egentligen av värdet hex 52. Till detta skall dock läggas hex 80 för att tala om att det är en strängvariabel. Därefter anropas subrutinen $DOE7, som letar rätt på var strängvariabeln är lagrad i minnet. Nu lagras dock inte strängen direkt på detta ställe utan de tre bitarna efter variabelnamnet anger strängens längd samt dess minnesposition (start). Denna position lagras i cellerna $FD–FE.

Vi vet nu dels var vårt tecken ligger ($FB–$FC dvs 251 och 252) och dels var strängen ligger som vi ska lagra de nya byten i. Den tidigare beskrivna omformningen sker mellan 5ECA och 5EE4. Först hämtas de nedersta byten i 8*8 teckencellen och roteras åt höger. Därvid lagras den minst signifikanta biten i carry och den roterar sedan in specialtecknet på analogt vis. Detta upprepas ett antal gånger med successivt minskade Y- och X-värden, varefter återhopp sker till BASIC genom RTS.

Instruktionerna PHP, PLP och NOP behövs egentligen inte här men har medtagits för att en 4 ggr expanderad kopia även ska kunna erhållas med vissa smärre modifieringar av programmet. Dessa beskrivs senare.

Forts på sidan 24

Att skriva program som subrutiner

Tänka sig att vi nått det åttonde avsnittet i Assemblerskolan. En hel del har vi lärt oss men mycket kvarstår, så låt oss se vad som mer döljer sig i 65xx:ornas mjuka värld.

Det första jag vill framföra är ett stort tack från Krulliga Krut & Co och deras anställda till alla läsare av Assemblerskolan för den hjälp dom fick av er i förra avsnittet. Nu flyter allt bra igen på företaget låter man hälsa.

I detta nr skall vi titta på hur man skriver sina program som subrutiner. Vi har ju redan använt oss av en subrutin när vi printat saker på skärmen ($FFD2). Men hur skriver man sådana och vad bör man tänka på.

ASSEMBLERSKOLAN del 8

För att få ett riktigt utbyte av ett sådant här resonemang så skall vi först titta på instruktioner som berör stacken. Som du kanske kommer ihåg så har vi hos 65:an ett register som alltid pekar på stackens nästa fria position. Vi kallar detta register stackpekaren eller SP. Du kommer med all säkerhet ihåg att stacken hos 6502:an alltid ligger på sida ett d v s mellan adresserna $0100–$01FF samt att stacken 'växer' mot lägre minne.

Detta sätt att lagra brukar man kalla en LIFO-struktur d v s Last In First Out (Sist In Först Ut) eftersom hämtning och ditläggning av data på stacken följer detta mönster. För att få oss en bild av hur detta ser ut så tittar vi på ett exempel.

Antag att vi har initierat stackpekaren så att den pekar på toppen av stacken och att vi sedan lagt dit $DE,$10,$FF och $F0 i tur och ordning. Stacken har då följande utseende

$DE	←botten av stacken	$01FF
$10		$01FE
$FF		$01FD
$F0		$01FC
	←stackpekare	$01FB

Observera att stackpekaren **inte** pekar på det sista elementet vi lade på stacken utan på **nästa fria position.** Av detta ser vi vad som händer när vi lägger ett element på stacken.

a) Lägg elementet i minnesposition som SP pekar på.

b) öka SP med ett.

Låt oss nu blocka bort ett element från stacken i vår tankemodell. Hur ser den ut efter det att vi har plockat bort toppelementet.

$DE	←botten av stacken	$01FF
$10		$01FE
$FF		$01FD
	←stackpekare	$01FC

Som du ser så har $F0 försvunnit. Stackpekaren pekar också nu på nästa fria position. Nu vet vi också vad som händer när man plockar bort ett element från stacken.

a) Minska stackpekarens innehåll med ett.

b) Läs innehållet i den position som SP pekar på nu.

EVIGT FÖRLORAD

Som du ser så sker aldrig någon förstörelse av data. Ty trots att vi plockade bort $F0 ovan så ligger $F0 lagrat i RAM vid $01FC. Denna position är dock densamma som den position vi kommer att lägga nästa element på stacken på, så nästa gång vi lägger något på stacken så är den för evigt förlorad. Det



bästa sättet att se det hela på är alltså att anse stackens toppvärde som förlorat när vi har hämtat det. Om vi nu ville så skulle vi kunna använda X-registret som en stackpekare. Vi initierar registret genom att ladda med $FF.

```
LDA #$FF
```

Innehållet i X är nu ett index till nästa fria position på stacken. För att lägga ett ackumulatorns innehåll på vår låtsasstack använder vi sekvensen

```
STA $0100,X    ;lägg (A) på stacken
DEX            ;peka på nästa fria pos
```

och för att hämta det översta elementet

```
LDA $0101,x    ;hämta toppelement
INX            ;glöm bort elementet
```

Hos 6502an finns redan två instruktioner som sköter detta åt oss automatiskt.

PLA (Pull Ackumulator)

Lägg ackumulatorns innehåll på stacken.

PHA (Push Ackumulator)

Hämta det översta stackelementet och lägg i ackumulatorn.

Eftersom SP också är ett åttabits register så skulle PLA och PHA kunna utförts av följande sekvenser om vi hade en LDA STA adressmod som kunde ske indexerad med SP. **Observera att nedanstående instruktioner inte existerar.**

```
;simulera PHA
    STA $0100,SP   ;lägg (A) på stacken
    DESP           ;SP=SP-1
;simulera PLA
    LDA $0101,SP   ;hämta toppelement
    INSP           ;SP=SP+1
```

Detta är precis vad som också sker.

Nu skulle vi kunna visa exempel på hur man använder instruktionen men eftersom den kommer att dyka upp i många framtida exempel så låter vi det anstå. Håll istället ögonen öppna och titta på hur och varför PHA och PHP används i rutiner.

Två ytterligare instruktioner existerar för att lägga eller hämta register till/från stacken. Man skulle kunna vänta sig att dessa skulle vara instruktioner för att hämta/lägga innehållet i indexregistren på stacken. Sådana finns inte. Här får man använda TYA, TAY, TXA och TAX och sedan PLA, PHA. D v s

```
            Lägg (Y) på stacken
TYA
PHA
            Hämta (Y) från stacken
PLA
TAY
            Lägg (X) på stacken
TXA
PHA
            Hämta (X) från stacken
PLA
TAX
```

Instruktionerna då?

PHP (Push status)

Instruktionen lägger statusregistrets innehåll på stacken. Detta kan vara användbart när man vid ett visst tillfälle vill spara flaggornas tillstånd för att kunna göra några flaggpåverkande operationer och sedan vid ett senare tillfälle ta tillbaks registret och testa på flaggorna.

PLP (Pull status)

Instruktionen tar tillbaka statusregistret från stacken. Och den används alltså oftast i kombination med PHP.

ÖNSKADE FLAGGOR

Observera att man t ex kan ladda ackumulatorn med önskad flaggställning och sedan lägga detta på stacken med PHA och ta tillbaks med PLP och vips så har man fått alla önskade flaggor i den konfiguration man vill ha dem. Detta är dock en metod som normalt inte behöver användas eftersom vi har di-

forts nästa sida

Mnemonic	Beskrivning	Flaggor	Adressmod	Kod	Antal bytes	Cykler	"X"	Objektbyte							
		N V – B D I Z C						7	6	5	4	3	2	1	0
PHA	Kopierar ackumulatorn upp till stacken	Påverkas ej	Implied	$48	1	3	–	0	1	0	0	1	0	0	0
PLA	Tar in översta stackelementet och placerar i ackumulatorn	N och Z	Implied	$68	1	4	–	0	1	1	0	1	0	0	0
PLP	Tar in översta stackelementet och placerar i statusregistret	N,V,B,D,I,Z,C	Implied	$28	1	4	–	0	0	1	0	1	0	0	0
PHP	Kopierar statusregister upp till stacken	Påverkas ej	Implied	$08	1	3	–	0	0	0	0	1	0	0	0
TSX	Kopierar stackpekaren till X-registret	N	Implied	$BA	1	2	–	1	0	1	1	1	0	1	0
TXS	Kopierar X-registret till stackpekaren	Påverkas ej	Implied	$9A	1	2	–	1	0	0	1	1	0	1	0
RTS	Return från Subrutin PC–1, läses in från stack och ett adderas till detta värde	Påverkas ej	Implied	$60	1	6	–	0	1	1	0	0	0	0	0
JSR	Hoppar till subrutin och fortsätter exekvering här. PC-1 sparas på stacken som returnadress	Påverkas ej	Absolut	$20	3	6	–	0	0	1	0	0	0	0	0
JMP	Hoppar till en ny position och fortsätter programexekveringen här	Påverkas ej	Absolut Indirekt	$4C $6C	3 3	3 5	0 1	0	1	X	0	1	1	0	0

Forts från sidan 21

"Assemblerskolan"

rekta instruktioner för ändamålet. Om man vill påverka många flaggor på samma gång så kan metoden vara att föredra. Kom då bara ihåg att kommentera en sådan operation noggrant eftersom den annars lätt blir en "frågeteckensoperation".

För att sätta stackpekaren till ett känt tillstånd använder man

TXS

Denna instruktion överför innehållet i x-registret till stackpekaren. D v s när man initierar stackpekaren, vilket normalt görs vid uppstart, så använder man vanligtvis

```
LDX #$FF
TXS
```

som alltså sätter hela stacken som fri.

Denna instruktion kan man även använda om man någongång har fått skräp på stacken som man vill bli av med. Inuti ett program skall den dock användas med största försiktighet av anledningar som vi snart skall se.

En instruktion för överföring åt andra hållet finns 'naturligtvis' också.

TSX

Flyttar innehållet i stackpekaren till x-registret. Denna instruktion är användbar när man vill läsa toppelementet på stacken utan att förstöra det, d v s ta bort det. Man använder sig då bara av

```
TSX
LDA $0101,X
```

och laddar då alltså ackumulatorn med toppelementet på stacken nondestruktivt.

SUBRUTINER

Så var det då subrutinerna.

Vi förutsätter här att alla vet vad en subrutin är för slags djur. Som vi sett så använder man

JSR Jump To Subroutin

för att göra ett "call' till en subrutin. Vi har t ex använt

JSR $FFD2

när vi velat printa ett tecken på skärmen. Vi har då sett att detta utförs i rutinen och att programmet sedan fortsätter vidare.

Hur går detta till? Låt oss anta att vi har en subrutin som ligger vid $2000 och att vårt huvudprogram ligger vid $1200. De båda programmen har följande utseende:

```
1200   LDA #$41     ;'a'
1202   JSR $2000
1205   BRK
och
2000   PHA          ;spar
2001   LDA #$0d     ;return
2003   JSR $FFD2    ;printa
2006   PLA          ;tillbaks
2007   JSR $FFD2    ;printa
200A   RTS
```

Programmen är medvetet dåligt kommenterade så att du själv kan träna på att sätta dig in i hur de fungerar. Gör det!

Vad händer när vi skriver:

G 1200

Det första steget är elementärt. Vi laddar ackumulatorn med ASCIIkoden för A.

Sedan gör vi subrutinanropet till$2000. Detta betyder helt enkelt att programmet gör ett hopp till adress $2000 och fortsätter att köra den kod som ligger här. Innan detta hopp utförs så måste det på något sätt spara information om varifrån hoppet skedde så att det kan hoppa tillbaka igen när subrutinen är utförd.

Den information som sparas är tillbakahopps-adressen−1 d v s i vårt fall sparas

$1205−1 = $1204

Var sparas då detta 16-bits tal?

Jo, naturligtvis på stacken! Stacken är det ställe där man snabbast kan lägga och hämta information. Alltså tar processorn högbytet ($12) och lägger det på stacken och sedan lågbytet ($04) och lägger också detta på stacken och efter det att detta är utfört så gör den hoppet till $2000. När vi kommer hit har alltså stacken utseendet:

```
$12
$04
        ← stackpekare
```

Här lägger vi själva ackumulatorns innehåll på stacken (ASCIIkoden för A) och stacken får nu utseendet

```
$12
$04
$41
        ← stackpekare
```

Vi laddar sedan ackumulatorn med return ($0d) och gör en subrutin till printrutinen $FFD2. Återigen tar processorn och spar återhoppsadressen −1 på stacken och denna får nu utseendet

```
$12
$04
$41
$20
$05
        ← stackpekare
```

I slutet av printrutinen finns en

RTS ReTurn from Subroutin

Denna instruktion tar de två översta byten som ligger på stacken. Adderar ett och lägger resultatet i programräknaren (PC). Eftersom PC alltid pekar på nästa instruktion som skall utföras så innebär detta att programmet kommer att fortsätta på den adress som vi nu har i PC. D v s i vårt fall så kommer vi att fortsätta vid

$2005+1 = $2006



stacken har fått utseendet

$12
$04
$41
← stackpekare

Processorn hittar instruktionen PLA här och den laddar ackumulatorn med toppelementet på stacken. D v s ackumulatorn får innehållet $41 och stacken får utseendet

$12
$04
← stackpekare

Vi gör ett nytt subrutinanrop till printrutinen och stacken får utseendet

$12
$04
$20
$09
← stackpekare

När vi når RTS i printrutinen tas alltså $2009 tillbaks från stacken. Ett adderas och resultatet lagras i PC. D v s programmet fortsätter med den instruktion den hittar vid

$2009+1 = $200A

Stacken har utseendet

$12
$04
← stackpekare

Här påträffar vi en till RTS. D v s $1204 tas tillbaks från stacken. Ett adderas. Resultatet lagras i PC. D v s programmet fortsätter vid $1205 och vi är 'hemma' igen. Här påträffas BRK och allt är slutfört.

Man måste som du ser tänka på att en subrutin inte lämnar data på stacken. Antag t ex att vi vid $C000 har följande subrutinanrop

```
C000    JSR $2000
C003    BRK
2000    LDA #$FF
        PHA
        RTS
```

Vid $C000 kommer vi då att ha en stack med utseendet

$C0
$02
← stackpekare

I subrutinen vid $2000 lägger vi $FF på stacken d v s den får utseendet

$C0
$02
$FF
← stackpekare

MED EFTERTANKE

När vi når RTS så tar denna in de två översta stackelementen. Den förutsätter att lågbytet kommer in först följt av högbytet. D v s den tar in $02FF. Adderar ett till denna adress och laddar PC med detta värde. Resultatet blir här alltså att vi får en 'return' till $0300 istället för $C003 som vi hade önskat.

Nu kanske du förstår varför man skall använda PHA/PLA med eftertanke. En tanke kanske slår dig är att man kan använda RTS för att göra ett hopp till en bestämd rad. T ex kan vi tänka oss att vi vill hoppa till $C000. Vi lägger då $BFFF på stacken med följande sekvens

```
2000    LDA #$BF
        PHA
        LDA #$FF
        PHA
```

stacken får nu utseendet

$BF
$FF
← stackpekare

Vi kör sedan en RTS och vips så har vi fått ett hopp till $C000. Den totala koden får alltså följande utseende

```
2000    LDA #$BF
        PHA
        LDA #$FF
        PHA
        RTS
```

Det skall direkt konstateras att detta är ett enormt klumpigt sätt att utföra det hela på eftersom det finns en instruktion som utför detta automatiskt.

JMP JUMP

Ovanstående kod skulle alltså kunnat ersättas med

```
2000    JMP $C000
```

Vad gör denna instruktion egentligen.

Den laddar helt enkelt PC med det absoluta tal som följer. D v s i fallet ovan så laddas programräknaren med $C000 och följaktligen så fortsätter programmet på denna adress. Instruktionen skulle alltså lika gärna ha mnenomicen LDPC d v s ladda PC absolut.

forts nästa sida

Forts från sidan 19
"Kopiera skärmen med printer 1526"

VARIABLER

PZ$ skapar utrymme för maskinkodsrutinen i form av en sträng om 49 bytes,
SY startadress för maskinkodsrutinen,
CM teckenminnets startadress,
SM skärmminnets startadress,
NH 1 eller 2 för 8*8 respektive 16*8 tecken,
NR antalet teckenrader på skärmen,
NC antalet teckenkolumner på skärmen,
PR$ sträng för lagring av de 8 bytes som representerar 8*8 teckenceller till printer,
PM$ ett antal blanka tecken som bestämmer vänstermarginalen vid utskriften. Här satt till 5 blanka,
SK+J aktuella tecknets skärmadress,
CA startadress för den aktuella 8*8 teckencellen som ska printas ut,
N$ sträng som representerar ett blankt printertecken. Används för att snabba upp utskriften genom att blanka tecken hoppas över. Sliter mindre på printern då.

ANVÄNDNING

Programmet visar i sin nuvarande utformning hur stjärnan ritas på skärmen och kopieras med printern. Andra grafiktillämpningar kan köras genom att lägga in dessa i raderna 230–260. Alternativt hoppa till rad efter subrutinen t ex 230 GOTO 1000.

I rad 110–210 initieras minne, maskinkod m m. Behövs inte superexpander så tag bort GRAPHIC-kommandona på rad 110. Om det som ska kopieras ligger i annat skärmminne och teckenminne så kan SM och CM sättas före anrop av subrutinen 280. Stryk i så fall raderna 290–320.

Upprepad figurritning kan ske genom nya anrop av subrutinen. Eventuellt behöver radframmatningen ändras om text ska skrivas mellan figurerna. Fyll inte ut skärmen med onödig information. Detta förlänger utskriftstiden. Skriv istället på vanligt sätt under figuren.

EXPANDERAD KOPIA

En ytmässigt fyra gånger större kopia kan erhållas med följande modifieringar:
Rad 170 sista värdet 7 byts ut mot 3 d v s LDX#$03.
Rad 190 sista värdet 234 byts ut mot 106 d v s ROR istället för NOP.
Rad 430 step 4 i stället för step 8.
Rad 460 sista printsatsen ändras till
PRINT#4,PP$+CHR$(14)
+CHR$(254) för expanderad utskrift i sidled.
Rad 470 ändras till PP$=PP$+"(2blanka)" istället för tidigare 1 blank.

Programlistning se sid 35

Drinkbar



Här är ett annorlunda spel för VIC 20 men som ändå är spännande och roligt. Spelet är gjort för blivande affärsmän med drömmar om ett eget företag.

Spelet heter DRINKBAR och är ett affärsspel med både glädje och sorg. Om du gillar att chansa och ta risker så är det här spelet något för dig.

Spelet är relativt enkelt, men det är svårt att nå bra resultat. Du gör din beställning på drinkar och bestämmer priset. Därefter skrivs några instruktionsrader och beställda drinkar ut på skärmen. Detta följs av vädret, vilket kan variera från åska och regn till hett.

Slutligen skrivs din förtjänst och den sammanlagda kassan ut på skärmen. Det gäller att göra så bra affärer som möjligt på ett visst antal omgångar.

Prog

Rad:
60
70
90–130
150
160–200
210–220
230
240–270
290
310–320
330–350

Forts från sidan 23
"Assemblerskolan"

Observera att en indirekt mod också existerar för JMP, denna har vi redan tittat på när vi gick igenom adressmoderna så leta fram detta nr om du är osäker på denna adressmod.

TUSENTAL

Detta får bli allt för denna gång. Vi vill dock avsluta med en liten uppmaning. Av de brev vi fått till redaktionen har vi märkt ett stort intresse för assemblerprogrammering på 6502an (6510). Detta betyder att det sitter tusentals människor över hela landet och skriver stora och små assemblerprogram. I samtliga av dessa program har man smårutiner som adderar, subtraherar, plottar punkter, visar dikmenyer, gör talkonverteringar o s v, o s v.

Som det är idag så sitter vi alla och knåpar och knäpper och i många fall gör samma kodningsarbete. Detta tänkte vi att VIC rapport skulle hjälpa oss att undvika. Om alla bestämmer sig för att skicka in kommenterade listningar av sina hjälprutiner så skall vi kontinuerligt publicera dessa. Till slut bör vi kunna få ett riktigt ordentligt bibliotek med assemblerrutiner och tillsammans så underlättar vi alltså programmeringsarbetet för varandra. Så sätt igång och överös oss med assemblerrutiner. Adressen är

Assemblerskolan
VIC rapport
Box 42054
126 12 Stockholm

Till den här adressen är du också välkommen att skriva om du har åsikter, önskemål eller något annat på hjärtat.
Go For It! □

– VIC 20



rambeskrivning:

rensar skärmen och sätter på volymen.
ändrar färg.
instruktionstext.
väntar på en tangentnedtryckning.
spelet börjar och frågar efter beställningsantal.
frågar om beställningen är för stor.
frågar efter priset.
ritar ut beställda drinkar och skriver instruktioner.
väntar på en tangentnedtryckning.
spelar upp en melodi.
bestämmer vädret.

360–410	framkallar åska.
420–490	skriver och ritar "molnigt".
500–580	skriver och ritar "hett".
590	pengar till grossisten.
610	brutto vid samtliga sålda.
620–650	bestämmer hur mycket pengar det blev sålt för.
660–670	räknar ut antalet sålda.
680–710	skriver ut vinsten och antalet sålda.
720–740	skriver ut summan och antalet omgångar som återstår.
750	testar om spelet är slut.
760–790	väntar på en tangentnedtryckning och börjar om.
800–820	skriver ut slutresultat.
850–870	nollställer resultatet och väntar på nedtryckt tangent.

Jyrki Honkanen



```
10 REM*****************
20 REM*  DRINK BAR    *
30 REM* BY J.HONKANEN*
40 REM* COPYRIGHT 84 *
50 REM*****************
60 PRINT"▒":W=8:POKE36878,15
70 POKE36879,8
80 Q=36879
90 PRINT"▒*** DRINK BAR ***   "
100 PRINT"▒SPELET GAR UT PA ATT"
110 PRINT"▒SAMLA IHOP SA MYCKET"
120 PRINT"▒PENGAR SOM MOJLIGT"
130 PRINT"▒PA 9 OMGANGAR"
140 PRINT:PRINT"▒▒▒TRYCK VALFRI TANGENT!"
150 GETA$:IFA$=""THEN150
160 PRINT"▒"
170 POKE36879,8
180 PRINT"▒ *** DRINK BAR ***   "
190 PRINT:PRINT"HUR MANGA DRICKA":PRINT
200 INPUT"VILL DU BESTALLA";A
210 IFA>87THENPRINT"▒▒▒▒▒▒SLUTA LARVA DIG !!"
220 IFA>87THEN190
230 PRINT:INPUT"PRIS KR/STYCK";B:PRINT"▒▒"
240 IFA=0THEN260
250 FORI=1TOA:PRINT"▒"TAB(I-1)"▒ ":NEXT
260 PRINT:PRINT"BRUTTO VID SAMTLIGA ":PRINT"SALDA BLIR";A*B;"KRONOR"
270 PRINT"▒▒GROSSISTPRISET AR 2.5 KR/ST"
280 PRINT:PRINT"▒TRYCK VALFRI TANGENT!"
290 GETA$:IFA$=""THEN290
300 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒WEATHER":POKE36878,15
310 FORI=1TO19:READX,Y:POKE36876,X:FORT=1TOY:NEXT:NEXT:RESTORE:POKE36876,0
320 PRINT"▒▒":POKE36879,25
330 S=INT(RND(1)*7):IFS=1THENJ=1:GOTO360
340 IFS=2ORS=3ORS=4ORS=5THENJ=2:GOTO430
350 IFS=6ORS=7THENJ=3:GOTO500
360 FORR=1TO4:POKE36879,8:POKE36879,25:POKE36879,8
370 POKE36877,220:FORL=15TO0STEP-1:POKE36878,L
380 FORM=1TO50:NEXTM
390 NEXTL:POKE36877,0
400 NEXTR
410 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒RAIN !!!!"
420 FORI=1TO3000:NEXT:GOTO590
```

Programmet fortsätter på sidan 44

Om FORTH – med finess

av Ola Johansson



Denna artikel, som är den femte i serien om FORTH i VIC rapport, ska ta upp problemet att skriva applikationer på ett "snyggt" sätt. Detta är nödvändigt om man ska kunna följa applikationernas struktur vid ett senare tillfälle.

Motståndare till FORTH brukar ofta säga att dess kod är näst intill oläslig. Tyvärr kan det stämma i vissa fall, när en programmerare inte skriver koden enligt ett visst mönster. FORTH har nämligen inte någon inbyggd kontroll av hur applikationstexten utformas, utan man kan i princip skriva hur som helst så länge man följer språkets regler om ordföljd m m. För att undvika att applikationerna blir svårlästa krävs det att man följer en del regler under programskrivandet. Några sådana nämndes i förbigående i förra artikeln, men eftersom detta är så viktigt så ska det behandlas närmare nu.

ENKLA KORTA DEFINITIONER

Det första man ska tänka på är att hålla sina definitioner korta. Om man gör det så kommer det samtidigt att bli lättare att följa deras struktur. En kort definition brukar vara ca en till fem rader lång (varje rad är på 40 tecken). Det värsta som kan hända är om man skriver en definition som fyller en hel skärm. Denna måste delas upp om den ska kunna förstås.

ETT BRA NAMN SÄGER ALLT

Det räcker nu inte med att dela upp applikationen i flera små ord, man måste också ge dem ett bra namn. Ett bra namn ska säga vad ett ord gör för något på ett kortfattat sätt. Om man dessutom håller sig till engelska (det språk som de ursprungliga orden är skrivna på) så blir dina FORTH-ord mer lättlästa. Jag själv brukar ibland skriva de sista orden på svenska så att det är lättare för en utomstående att använda dem.

Eftersom ordnamnen ska vara korta och förklarande har man infört en del standardtecken med olika betydelser som man kan använda vid namngivningen. En del sådana specialtecken har ni redan stött på, som t ex @ (som betyder "Fetch", hämta) och ! (som betyder "Store", lagra). En lista över några vanliga specialtecken finns i tabell 1.

Tabell 1

SPECIALTECKEN I ORDNAMN

Tecken före namn

tecken	innebörd	exempel
!ord	"store", lagra	!DATA !CSP
@ord	"fetch", hämta	@DATA
(ord)	intern del av "ord"	(CROSSREF)
'ord	adress till "ord"	'DEFAULT
+ord	plus, bättre "ord"	+BUF +EDITOR
/ord	division	/CHECK
?ord	test av "ord", gör "ord" om parametern är "sann".	?EMIT ?DISC
−ord	subtraktion, ta bort "ord"	– –TRAILING
Cord	tecken, 1 byte	C@
Dord	dubbelprecision	D.

Tecken efter ordnamn

tecken	innebörd	exempel
ord"	text kommer efter "ord"	NAMN" ."
ord,	lagra i ordlistan	C,
ord?	kolla något, ge sant/falskt	MARK?

Detta är ett urval av tecken, som är bra att använda för att hålla ordnamnen korta samtidigt som de säger mycket. Exempelvis kan man tala om att något lagras bara genom att sätta ett utropstecken före (eller efter) ordnamnet.

SPARA INTE PÅ MELLANSLAGEN

För att ytterligare förtydliga dina program ska du vara frikostig med mellanslag. Dessa används för att dela upp varje definition i fraser. Som du vet krävs minst ett mellanslag mellan varje ord, men om du dessutom lägger dubbla mellanslag mellan varje logisk fras i definitionen förtydligar du texten avsevärt. Olika logiska fraser kan t ex vara:

```
10 0 DO
? TERMINAL UNTIL
?SHIFT IF
0< WHILE
YEAR @
```

Känslan för att avgöra vad som är en "logisk fras" kommer allt eftersom man programmerar i FORTH, men som grundregel kan man säga att det är bättre att frasindela än att låta bli.

När du använder den inbyggda assemblern så ska du ha två mellanslag mellan **varje** assemblerinstruktion, som t ex

```
XSAVE STX, eller
NEXT JMP, .
```

Det finns även några ställen där du ska ha minst **tre** mellanslag, och det är efter en definitions namn och i början av varje ny rad i en definition. När jag definierar ord brukar jag bara ha definitionens namn följt av stackkommentaren på en rad. Sedan byter jag rad och delar på så sätt upp varje definition i en namndel och en koddel.

TOMMA RADER ÄR LIKA VIKTIGA SOM FULLA

Förutom att man ska lägga in många mellanslag mellan sina ord, så är det minst lika viktigt att ha tomma rader mellan ordens definitioner. Förutom att det ökar läsbarheten så ger det även en möjlighet att utöka orden vid senare tillfällen. I och med att man lägger in flera tomma rader så minskar man samtidigt antalet ord som får plats på varje skärm. Man ska nämligen aldrig trycka in så mycket som möjligt, utan hellre sprida ut applikationen på flera skärmar. Samtidigt kan man försöka gruppera om orden, så att ord som inte har något gemensamt hamnar på olika skärmar.

När jag ändå är inne på rader så kan det vara på sin plats att nämna första raden på varje skärm. Den ska nämligen bara bestå av en kommentar, vilken ska innehålla en kort beskrivning av skärmens innehåll samt datumet då skärmen skrevs. Det kan också vara lämpligt att ha med författarens signatur eller något liknande, så man vet vem som är "skyldig" till innehållet.

EN BRA STACK ÄR EN REN STACK

Den något kryptiska rubriken ovan syftar på hur orden ska påverka stacken. De flesta ord behöver ett eller flera tal på stacken (så kallade parametrar) för sin funktion. Det kan exempelvis gälla ordet + (plus), som behöver två tal vilka ska adderas. När additionen är gjord så lämnas summan kvar på stacken. Det viktiga i sammanhanget är att **bara** summan och inget annat lämnas kvar. Detta kallas för att orden förstör sina inparametrar. **Alla** ord **ska** förstöra sina parametrar.

Om man absolut vill använda samma tal senare kan man använda de olika kopieringsord som ger möjlighet att placera en extra kopia av talet på stacken. För att ge ett enkelt exempel på detta antar jag att vi har definierat orden XY@ (som ger ett koordinatpar), !DATA (som lagrar koord. i en tabell, för senare utritning) och PLOT (som ritar en punkt på en given koordinat). Ett ord som gör att man kan rita på skärmen skulle då kunna vara:

```
: DRAW
   BEGIN XY@ 2DUP !DATA
   PLOT ?TERMINAL UNTIL ;
```

Man kan då falla för frestelsen att placera 2 DUP inuti !DATA så att ordet ser "snyggare" ut:

```
: DRAW
   BEGIN XY@ !DATA PLOT
   ?TERMINAL UNTIL ;
```

!DATA förstör då inte sina inparametrar (koordinaterna) utan lämnar dem på stacken (till PLOT). Problem kommer då att uppstå ifall man vill använda !DATA någon annanstans. Då måste ett 2DROP följa varje anrop, så att stacken töms från parametrarna. Även detta uttryck har fått en benämning; ordet har blivit beroende av sin omgivning. Ord ska **inte** vara beroende av sin omgivning.

TOP–DOWN, ETT SÄTT ATT PROGRAMMERA

Ett fel, som kan förekomma hos de som just har börjat med FORTH, är att de börjar med att skriva de "nedersta" orden, d v s de ord som sedan ska anropas av andra ord. Detta kan ofta resultera till att man anstränger sig för att "passa in" dessa ord i de senare orden. Som ni förstår kan inte detta resultera i bra program. Ett ännu värre fel är att man inte använder alla ord, utan man står där med en mängd ord som inte gör annat än tar upp en massa plats. Även detta leder förstås till dåliga program.

För att undvika detta bör (läs "ska") man använda en metod som har fått namnet "top–down programmering". Detta innebär att man skriver sin applikation genom att först definiera sitt problem, sedan dela upp detta i mindre problem och fortsätta med det tills hela problemet är löst.

Det kan sammanfattas i några punkter:

1. Definiera problemet:
   Tänk efter vad applikationen egentligen ska göra och bestäm vilka saker som behöver matas in (indata) och vad som ska komma ut (utdata).
2. Strukturera problemet:
   Dela upp problemet i delproblem, som vardera löser en logisk del av det som ska lösas.
3. Formulera problemet i FORTH:
   Skriv definitionen genom att ge varje delproblem ett namn och placera dessa i din applikation.
4. Upprepa tills applikationen är klar:
   Genom att dela upp varje delproblem i än mindre problem kommer du till slut till en punkt då nya ord inte behöver definieras. Då är applikationen klar.

Forts på sidan 28

Forts från sidan 27
"Om FORTH..."

Under hela programskrivningen testar du varje ord, så att de fungerar som du har tänkt dig. Varje ord du skriver ska också innehålla en kommentar om hur de påverkar stacken. Om du håller orden korta ska ytterligare kommentarer inte behövas. Ett exempel på en kommentar:

```
: /CHECK ( täljare nämnare -
  kvot )
  -DUP IF / THEN ;
```

DOKUMENTERING

Under programskrivandets gång, eller då applikationen är klar, ska du sammanställa en dokumentation av ditt arbete. En bra dokumentation bör innehålla en lista över de ingående orden samt en förklarande text, som beskriver applikationens funktion. En lista över de ingående orden brukar kallas för en **korsreferenslista.** För att underlätta skapandet av en sådan lista ska jag senare i artikeln presentera en applikation kallad CROSSREF. För att beskriva applikationen kan man använda en metod som är uppfunnen av Forth Inc., nämligen de så kallade **skuggblocken.**

```
SCR # 30
( CROSSREF - KORSREFERENS          OJ-840825)

EDITOR DEFINITIONS

: REPORT
                              ( ROW SCR -- *)
  2DUP (LINE)  CR  5 SPACES  TYPE  SWAP
  4 .R  4 .R  ;

: MARK?
                              ( ADR -- FLAG *)
  1-  DUP  1-  C@  42 =  SWAP  C@
  41 =  AND  ;

: (CROSSREF)
                              ( SCR -- *)
  25 0 DO  FORTH I EDITOR  OVER (LINE)
  +  MARK? IF  FORTH I EDITOR  OVER
  REPORT  THEN  LOOP  DROP  ;

: CROSSREF
                              ( SCR1 SCR2 -- *)
  1+ SWAP DO  FORTH I EDITOR
  (CROSSREF)  CR  LOOP  ;

FORTH DEFINITIONS  ;S
```

SKUGGBLOCK

Det finns inget skumt över dessa block, utan de är faktiskt utmärkta hjälpmedel för dokumenteringen. De heter skuggblock eftersom de fungerar som en "skugga" till ett vanligt programblock, eller skärm. Skuggblocken innehåller inte någon programtext som kan laddas in med ordet LOAD. Istället finns där en beskrivning av varje ord som är definierat i det block som skuggblocket är "skugga" till. Med andra ord: Varje skuggblock ligger ett bestämt antal block efter programblocket. På de båda "diskmaskinerna" 1540 och 1541 kan det vara lämpligt att placera skuggblocken 100 block efter programblocken. Programblock 17 har då skuggblock 117, nr 42 har ett skuggblock med nummer 142. När man har skrivit sin applikation kan man sedan ta ut en lista (om man har en skrivare) där programblock och skuggblock står bredvid eller direkt efter varandra. På detta sätt har man fått en enkel men tydlig dokumentation. Detta kombineras givetvis med en korsreferenslista, som ytterligare förtydligar dokumentationen.

```
SCR # 130
( SKUGGBLOCK: CROSSREF             OJ-840825)

=> CROSSREF ANVÄNDS FÖR ATT SKAPA EN
 KORSREFERENSLISTA TILL EN APPLIKATION.
FÖR ATT DETTA SKA VARA MÖJLIGT MÅSTE DET
SOM ÖNSKAS TILL LISTAN STÅ PÅ EN RAD
VARS SISTA TVÅ TECKEN ÄR:  *)

=> CROSSREF  ( SCR1 SCR2 -- )
 ANROPAR (CROSSREF) MED SKÄRMNUMMER SOM
ARGUMENT.

=> (CROSSREF)  ( SCR -- )
 LETAR EFTER *) MED HJÄLP AV  MARK?  OCH
ANROPAR  REPORT  VID FÖREKOMST AV DETTA.

=> MARK?  ( ADR -- )
 SE OVAN.

=> REPORT  ( ROW SCR -- )
 MED RAD- OCH SKÄRMNUMMER SOM ARGUMENT
SKRIVS RADEN FÖLJT AV RAD- OCH SKÄRM-
NUMMER UT PÅ SKÄRM, ELLER AKTUELL UT-
ENHET.
```

CROSSREF – ETT EXEMPEL

Förutom att presentera ett användbart hjälpmedel för skapande av korsreferenslistor är detta tänkt att visa hur en "typisk" programutveckling kan tänkas att se ut när man använder FORTH.

Följ med i programskärmen (nr 30) för att se hur varje ord definie-

ras. Observera att jag börjar med att definiera CROSSREF, trots att det måste skrivas in **efter** de övriga orden. Det är detta som kallas för "top–down"-programmering.

CROSSREF ska fungera så att man anger ett intervall av skärmar och får en korsreferenslista utskriven på det som finns i intervallet. CROSSREF ska leta efter rader vars sista två tecken är *) och sedan skriva ut dessa rader följt av radens och skärmens nummer. Det kan exempelvis se ut så här:

: CROSSREF (SCR1 SCR2 – – *)
17 30

där 17 är radnumret och 30 är skärmnumret.

Man kan då tänka sig att CROSSREF måste bestå av någon form av programslinga, som stegar sig igenom skärm efter skärm i det givna intervallet. Om man använder DO LOOP, som är lämplig i detta fall, måste man tänka på att addera ett till sista skärmnumret. Detta beror på att räknaren ökas i slutet av slingan och slingan avslutas då detta värde har uppnåtts eller passerat slutvärdet. I enlighet med det som sagts tidigare definierar vi CROSSREF så att den successivt ger ett skärmnummer till ett annat ord, tills intervallet är täckt. Detta ord kallar vi för (CROSSREF), där parentesen betyder att detta ord anropas inuti CROSSREF och inte är gjort för att användas direkt.

(CROSSREF) i sin tur ska söka igenom en skärm i taget, för att leta efter rader som slutar med*). Detta måste också vara en slinga, lämpligen DO LOOP, som tittar i slutet av varje rad och rapporterar om raden har *) som avslutning. Eftersom dessa ord definieras i EDITOR-vokabuläret så måste man "byta" till FORTH tillfälligt när ordet I anropas i slingan (du kommer väl ihåg att ett ord med namnet I finns i både EDITOR- och FORTH-vokabulären?)

För att leta efter *) definierar vi ett ord som hittar *) i två på varandra följande adresser och för att rapportera eventuella rader definierar vi ett ord kallat REPORT, som skriver ut raden med TYPE och sedan rad- och skärmnummer med ordet .R (.R skriver ett tal högerjusterat i ett fält, som i detta fall består av 4 tecken.)

När man har tänkt ut ordens definitioner, och eventuellt skrivit ned dessa på ett papper, så kan man börja skriva in orden i en skärm. Nu ska man dock skriva in orden från "botten" och "uppåt", eftersom FORTH alltid använder ord som har definierats tidigare och som redan finns i ordlistan. När sedan skärmen laddas in kommer dessa fyra ord att kompileras in i EDITOR-vokabuläret.

NÄSTA ARTIKEL

I nästa artikel kommer jag bland annat att skriva om hur man kan göra sina egna kompilatorer i FORTH. Det är en av de mest kraftfulla sakerna hos detta språk. Det enda som återstår av denna artikel är sammanfattningen, så tills nästa gång: Lycka till med FORTH-programmeringen och jag hoppas ni har nytta av CROSSREF!! □

SAMMANFATTNING

Nedan har jag försökt sammanfatta det som nämnts i artikeln, så att du lätt kan överblicka de olika regler som bör följas vid programskrivning i FORTH.

- **Håll definitionerna korta.**
- **Ge orden namn som är korta och som talar om precis vad de gör för något.**
- **Dela upp orden i fraser, skriv tre mellanslag efter en definitions namn och i början av varje extra rad av en definition.**
- **Fyll inte skärmarna med för mycket definitioner utan ha många tomma rader för läsbarhet och framtida expansion.**
- **Alla ord ska förstöra sina inparametrar, vilket gör orden oberoende av sin omgivning.**
- **Skriv dina program enligt "top down"metoden.**
- **Dokumentera dina program med skuggblock och korsreferenslistor.**
- **Skuggblock ligger 100 block efter sina programblock.**
- **Utnyttja CROSSREF för att dokumentera dina applikationer.**

TJÄNA PENGAR PÅ SINA EGNA PROGRAM



Alla har vi nog någon gång gjort ett program som vi har blivit jättenöjda med. Då har vi även tänkt att det här skulle man sälja och tjäna pengar på. Men ofta så stannar det där, någon kanske sätter ut en liten annons i någon tidning. Vad de flesta inte gör, är att skicka en kopia av programmet till något s k mjukvaruföretag. Alltså ett företag som endast eller delvis säljer program för olika datorer.

Och varför gör man inte det? Kan man ju fråga sig. Kanske beror det på att programmet är skrivet i BASIC eller att man tror att DOM är väl inte intresserade av mina program.

Forts på sidan 44

Forts från sidan 17
"Tävlingsresultat"

3:e pris Nyttoprogram VIC 20

Tävlingsbidrag:
TABELL–20 av Thomas Hedlund
Utrustning:
VIC 20, kassett (printer)
***Betyg:* 3**

Omdöme: Ett fungerande program inom ett originellt område. Mycket användbart för den kategori som den vänder sig till.

Programmet hanterar tabeller i samband med serieidrott. Programmet hanterar upp till 12 lag samt räknar ut poäng och målskillnader. Lagen sorteras i rätt ordning och skrivs ut på skrivaren om så önskas. Grafiken är bra men ljud saknas.

Författarens egna förklaringar till programmet:

Tabell–20 är precis vad det tycks vara; en tabell till VIC 20. Man kan ha upp till 12 lag med, och varje lags namn, spelade matcher, målskillnad och poäng hålls reda på. Dessutom räknar datorn hur många matcher som har spelats, om man spelar en serie eller så.

När man skriver in resultat, räknar datorn automatiskt ut poäng och målskillnad, och sorterar lagen efter i första hand poäng, i andra hand målskillnad, och i tredje och sista hand, lagnamnet.

Tabellinnehållet kan antingen sparas på band eller diskett, eller skrivas ut på printer. Utskriften är gjord för en MPS–801 och därför kan en förklaring av CHR$-koderna vara på sin plats.

CHR$(8) = Grafikutprintningstecknet. Gör så att printern inte matar fram en hel rad utan

Fuel Lander, del 2

```
90 DATA1,3,7,15,31,63,127,255,128,192,224,240,248,252,254,255,255,255,255,255,25
5,255
100 DATA255,255,128,192,128,192,232,252,254,255,128,192,128,192,192,226,246,255
110 DATA128,192,224,240,240,224,192,128,1,3,7,15,15,7,3,1,,,,,,,,10,94
120 DATA,,,,,,68,110,,,,,,16,58,126,1,3,1,3,3,71,111,255,255,252,254,252,248
130 DATA184,144,128,255,63,127,63,31,31,9,1,,,,16,16,,,,24,24,24,60,231,255,126
140 DATA153,176,112,120,239,239,120,112,176,153,126,255,231,60,24,24,24,13,14,30
,247
150 DATA247,30,14,13,24,60,102,255,189,255,66,66,66,66,255,189,255,102,60,24
180 SYS7087:PRINT"(HOME)(DOWN)(DOWN)(DOWN)(DOWN)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(R
GHT)(RVS)(YEL)GAME(RGHT)OVER(DOWN)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGH
T)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(BLU)PRESS(RGHT)FIRE";
185 PRINT"(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOW
N)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(OFF)YG(RGHT)(RVS)(CYN)O"
190 IFPEEK(0)<128THEN190
198 PRINT"(HOME)(DOWN)(DOWN)(DOWN)(DOWN)                              (DOWN)
           (DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOW
N)(DOWN)(DOWN)"
199 PRINT"(RGHT)(RGHT)(RGHT)(RGHT)(RVS)(CYN)O          (UP)":SC=0:N=7
200 POKE36878,15:PRINT"(UP)(RGHT)(RGHT)(RGHT)(CYN)AAAAAAAAAAAAAA":A=7749:B=0:C=2
6:G=1:H=16:I=8061:L=250:M=220
203 N=N-1:IFN=1THENPRINT"(UP)(UP)":AB=0:GOTO42
204 J=29:POKE8050,32:POKEA+30720,1:POKEA,26:K=INT(RND(1)*7)+1:GOTO390
205 SC=K*100+SC+H*10:PRINT"(BLU)HIJ(RVS)(CYN)"SC"(UP)":H=16:B=0:POKE8050,32
206 PRINT"(UP)(RGHT)(RGHT)(RGHT)(CYN)AAAAAAAAAAAAAA":K=INT(RND(1)*7)+1:IFI=8061T
HENI=8050:J=30:A=8039:GOTO390
207 IFI=8050THENI=8061:J=29:A=8072:AB=AB+10:POKE37879,AB:IFAB>77THENAB=70
208 PRINT"(RVS)(YEL)(UP)(UP)(UP)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)LEVEL=
"AB/10+1"(DOWN)(DOWN)":GOTO390
210 IFPEEK(0)>127THENB=0
215 PRINT"(UP)(UP)(BLU)YG(RVS)(CYN)"N-1"(DOWN)":POKE36875,130+AB:IFPEEK(0)=8THEN
B=1:C=26
220 IFPEEK(0)=4THENB=-1:C=28
230 IFPEEK(0)=1THENB=-22:C=25
240 IFBEEK(0)=2THENB=22:C=27
250 IFPEEK(A+B)=32ORPEEK(A+B)=24ORPEEK(A+B)=26ORPEEK(A+B)=27ORPEEK(A+B)=25THEN29
5
255 IFPEEK(A+B)=JORPEEK(A+B)=28THEN295
260 POKE36875,0
270 POKEA,32:POKE36877,255:FORD=15TO0STEP-.15
275 POKE36864,13:POKE36878,D:POKE36864,12:NEXT:POKE36877,0:GOTO200
295 IFPEEK(A+B)=JTHENPOKE8061,32:GOTO205
300 POKE36875,M
304 IFB=0THENPOKE30720+A,1:POKEA,C:GOTO320
305 POKEA,32:A=A+B:POKEA+30720,1:POKEA,C
320 POKE7798,24:POKE7846,24:POKE7734,24:POKE7859,24:POKE7810,24:POKE7730,24:POKE
7865,24
340 IFG=1THENPRINT"(UP)(CYN)"TAB(H);"B":G=2:GOTO210
350 IFG=2THENPRINT"(UP)"TAB(H);"C":G=3:GOTO210
360 IFG=3THENPRINT"(UP)"TAB(H);"D":G=4:GOTO210
365 IFH=3THENB=22:PRINTTAB(2);"(UP)(BLU)(RGHT) ":L=L-5:M=M-5:POKE36875,L:GOTO250
370 IFG=4THENH=H-1:PRINT"(UP)"TAB(H);"A ":G=1:GOTO210
390 POKE30720+I,K:POKEI,J:GOTO210

READY.
```

TABELL–20

```
READY.

5 DIM LA$(12),SM%(12),MS%(12),PO%(12):CR$=CHR$(13):POKE36879,218
20 PRINT"(CLR)(BLK)":FORI=1TO21:PRINT"*";:NEXT:PRINT
50 PRINT"*PL NAMN    SM  MS PO*":PRINT"*";:FORI=2TO20:PRINT"—";:NEXT:PRINT"*"
70 FORI=1TOMA:PRINT"*";MID$(STR$(I),1-(I>9));".";LA$(I);TAB(11+(SM%(I)>9))SM%(I)
;
85 PRINTTAB(15+(MS%(I)>9)+(MS%(I)<-9))MS%(I);"(LEFT)";
90 PRINTTAB(18+(PO%(I)>9))PO%(I);"(LEFT)";TAB(20)"*":NEXT
100 FORI=1TO21:PRINT"*";:NEXT:PRINT
120 PRINT"(RVS)    (BLU)(OFF)B(BLK)(RVS)ORJA OM                        (BLU)(OFF)S(RVS)(BLK
)KRIVA IN RESULTAT ";
145 PRINT"    (BLU)(OFF)F(BLK)(RVS)IL   MATCH(RVS)"TY"(LEFT)                 (BLU)(OFF)F1
(RVS)(BLK) SKARMUTSKRIFT    (OFF)";
160 GETA$:IFA$="B"THEN200
170 IFA$="S"THEN600
185 IFA$="F"THEN1491
187 IFA$="(F1)"THENGOSUB60000
190 GOTO160
200 PRINT"(CLR)(RGHT)(RGHT)TANK PA ATT ALLT SOM  AR INSKRIVET I          TABELLEN
RADERAS."
230 PRINT"(DOWN)(RGHT)(RGHT)VILL DU (RVS)F(OFF)ORTSATTA?"
240 GETA$:IFA$=""THEN240
250 IFA$<>"F"THEN20
310 PRINT"(CLR)(DOWN)(DOWN)(RGHT)HUR MANGA LAG/SPELARE SKALL VARA MED":CLR
330 INPUT"(RGHT)(RGHT)(RGHT)(2-12)";MA:IFMA>12THENMA=12
335 IFMA<2THENMA=2
340 DIMLA$(12),MS%(12),SM%(12),PO%(12)
350 FORI=1TOMA:PRINT"NAMN PA DELTAGARE":PRINTI;:INPUTLA$(I):LA$(I)=LEFT$(LA$(I),
6):NEXT
380 PRINT"(CLR)(DOWN)(DOWN)(DOWN)(RGHT)HAR NAGON/NAGRA":INPUT"(RGHT)MATCHER SPEL
ATS";A$:IFLEFT$(A$,1)<>"J"THEN20
410 FORI=1TOMA:PRINT"(CLR)(DOWN)ANTAL SPELADE MATCHER":PRINT"FOR "LAG$(I);:INPUT
SM%(I)
430 PRINT"MALSKILLNAD FOR ":PRINTLA$(I);:INPUTMS%(I):PRINT"POANG FOR "LA$(I);:IN
PUTPO%(I)
450 NEXT:GOTO760
600 PRINT"(CLR)":FORI=1TOMA:PRINTI;LA$(I):NEXT:INPUT"NR PA HEMMALAG";H
610 IFH<1ORH>MATHEN20
640 TY=TY+1:INPUT"NR PA BORTALAG";B
650 PRINT"(CLR)(DOWN)(DOWN)DA AR MATCHEN SOM HAR SPELATS":PRINTLA$(H)"-"LA$(B):P
RINT"STAMMER DET (J/N)"
680 GETA$:IFA$="N"THEN20
681 IFA$="J"THEN 690
683 GOTO 680
690 PRINT"MAL FOR "LA$(H);:INPUTMH:PRINT"MAL FOR "LA$(B);:INPUTMB
710 SM%(H)=SM%(H)+1:SM%(B)=SM%(B)+1:MS%(H)=MS%(H)+(MH-MB):MS%(B)=MS%(B)+(MB-MH)
740 PO%(H)=PO%(H)-(MB=MH)-2*(MH>MB):PO%(B)=PO%(B)-(MB=MH)-2*(MB>MH)
```

bara nästan, så att grafiktecken kan 'gå in' i varandra för en snyggare utskrift. CHR$(8) kan helt uteslutas.

CHR$(14) = Dubbelt så breda tecken. Önskas normala tecken, ändra CHR$(14) till CHR$(15).

CHR$(145) = 'Cursor up'-tecknet. Betyder att teckenmode stora bokstäver/grafik användes. Om teckenmode stora/små bokstäver vill användas, ändra CHR$(145) till CHR$(17).

CHR$(34) = Snuff (citationstecken).

CHR$(18) = Reverse on.

CHR$(146) = Reverse off.

Tabellen borde rymmas i en oexpanderad VIC 20, men om den inte gör det, uteslut band- eller disk-delen. □

```
760 FORI=1TOMA-1:FORJ=I+1TOMA:IFPO%(I)>PO%(J)THEN870
790 IFPO%(I)=PO%(J)THEN850
800 X=MS%(I):MS%(I)=MS%(J):MS%(J)=X:X=SM%(I):SM%(I)=SM%(J):SM%(J)=X
820 X=PO%(I):PO%(I)=PO%(J):PO%(J)=X:X$=LA$(I):LA$(I)=LA$(J):LA$(J)=X$
840 GOTO 870
850 IFMS%(I)=>MS%(J)THEN870
860 GOTO800
870 NEXT:NEXT:GOTO20
1491 PRINT"(CLR)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)                    (RVS)S(OFF)PARA":PRINT"
     (RVS)H(OFF)AEMTA"
1493 GET A$:IF A$="S"ORA$="H"THEN1496
1495 GOTO 1493
1496 PRINT"(DOWN)          (RVS)B(OFF)AND                              (RVS)D(OFF)ISK"
1497 GETB$:IFB$="D"THEN2500
1498 IFB$="B"THEN1500
1499 GOTO1497
1500 IFA$="S"THEN1505
1503 IFA$="H"THEN2000
1505 INPUT"FILNAMN";FI$:IFFI$=""THEN20
1510 OPEN1,1,1,FIL$:PRINT#1,MA
1520 FORI=1TOMA:PRINT#1,LA$(I)
1540 PRINT#1,SM%(I)
1550 PRINT#1,MS%(I)
1560 PRINT#1,PO%(I):NEXT
1565 PRINT#1,TY:CLOSE1:GOTO20
2000 INPUT"FILNAMN";FI$:IFFI$=""THEN20
2010 OPEN1,1,0,FI$:INPUT#1,MA
2030 FORI=1TOMA:INPUT#1,LA$(I)
2050 INPUT#1,SM%(I)
2060 INPUT#1,MS%(I)
2070 INPUT#1,PO%(I):NEXT
2080 INPUT#1,TY:CLOSE1:GOTO20
2500 INPUT"FILNAMN";FI$:IFFI$=""THEN20
2505 IFA$="S"THEN2510
2507 IFA$="H"THEN2600
2510 OPEN15,8,15,"IO":OPEN2,8,2,"0:"+FI$+",S,W":GOSUB3000:IFENTHEN2990
2515 PRINT#2,STR$(MA):GOSUB3000:IFENTHEN2990
2520 FORI=1TOMA:PRINT#2,LA$(I):GOSUB3000:IFENTHEN2990
2521 PRINT#2,STR$(SM%(I)):GOSUB3000:IFENTHEN2990
2522 PRINT#2,STR$(MS%(I)):GOSUB3000:IFENTHEN2990
2524 PRINT#2,STR$(PO%(I)):GOSUB3000:IFENTHEN2990
2530 NEXT:PRINT#2,STR$(TY):GOTO2990
2600 OPEN15,8,15,"IO":OPEN2,8,2,"0:"+FI$+",S,R":GOSUB3000:IFENTHEN2990
2610 INPUT#2,MA:GOSUB3000:IFENTHEN2990
2620 FORI=1TOMA:INPUT#2,LA$(I):GOSUB3000:IFENTHEN2990
2621 INPUT#2,SM%(I):GOSUB3000:IFENTHEN2990
2622 INPUT#2,MS%(I):GOSUB3000:IFENTHEN2990
2624 INPUT#2,PO%(I):GOSUB3000:IFENTHEN2990
2630 NEXT:INPUT#2,TY
2990 CLOSE2:CLOSE15:GOTO20
3000 INPUT#15,EN,EM$,ET,ES:IFEN=0THENRETURN
3010 PRINT"(DOWN)"EN,"(RVS)"EM$"(OFF)",ET;ES:FORT=1TO1500:NEXT:RETURN
60000 SI$=CHR$(14):BS$=CHR$(8):RV$=CHR$(18):RO$=CHR$(146):QT$=CHR$(34)
60030 MF$=CHR$(145):VR=PEEK(648)*256:OPEN4,4:PRINT#4:FORCL=0TOMA+4:QF=0:AS$=MF$:
FORRO=0TO21
60060 SC=PEEK(VR+22*CL+RO):IFSC=34THENQF=1-QF
60080 IFSC<>162THEN60110
60090 QF=1-QF:IFQF=1THENAS$=AS$+RV$+QT$:GOTO60170
60100 AS$=AS$+QT$+RO$:GOTO60170
60110 IFQF=1AND(SC>=128)THENSC=SC-128:GOTO60130
60120 IFSC=>128THENSC=SC-128:RF=1:AS$=AS$+RV$
60130 IFSC<32ORSC>95THENAS=SC+64:GOTO60160
60140 IFSC>31ANDSC<64THENAS=SC:GOTO60160
60150 IFSC>63ANDSC<96THENAS=SC+32:GOTO60160
60160 AS$=AS$+CHR$(AS):IFRF=1THENAS$=AS$+RO$:RF=0
60190 NEXTRO:IFQF=0THENPRINT#4,SI$AS$;:GOTO60210
60200 PRINT#4,SI$AS$;QT$;
60210 IFCL<4THENPRINT#4,BS$;
60211 PRINT#4:NEXTCL:PRINT#4:CLOSE4:RETURN

READY.
```

3:e pris Nyttoprogram VIC 64

***Tävlingsbidrag:*DEMO**
av Christer Harrysson
***Betyg:* 3,5 + 0,5 för originalitet**

Omdöme: Detta är ett mycket bra program om än med begränsat användningsområde. Programmet visar vad en VIC 64 kan göra vad gäller ljud och grafik. Programmet är bra gjort i sin helhet.

Programidén är originell. Grafiken är bra gjord och mycket instruktiv för den som inte känner till VIC 64ans grafik. Ljuddemonstrationen är också av bra kvalité samt ger för den ovetande en bra inblick i 64ans ljudmöjligheter.

DEMO

```
READY.

7 FORZ=54272TO54296:POKEZ,0:NEXT
10 FORA=1TO40:S$=S$+"(RGHT)":NEXT:FORA=1TO25:N$=N$+"(DOWN)":NEXT:GOSUB155
20 POKE53280,2:POKE53281,1:PRINT"(CLR)":POKE53275,255
21 B=0:FORA=0TO39:POKE1024+A,160:POKE55296+A,B
23 POKE2023-A,160:POKE56295-A,B:B=B+1:IFB=16THENB=0
25 NEXT
26 FORA=1464TO1704STEP40:POKEA,160:POKEA+54272,1:POKEA+1,160:POKEA+54273,1
27 POKEA+2,160:POKEA+54274,1:POKEA+3,101:POKEA+54275,0:NEXT
28 PRINT"(HOME)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(RGHT
)(RGHT)(RGHT)/(UP)/‾‾‾\(DOWN)\"
29 PRINT"(HOME)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN
)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(RGHT)(RGHT)(RGHT)\(DOWN)\___/(UP)/(H
OME)"
30 READA$:IFA$="*"THEN110
35 PRINTLEFT$(A$,1):A$=RIGHT$(A$,LEN(A$)-1)
40 FORA=39TO((40-LEN(A$))/2)STEP-1:PRINTLEFT$(S$,A)LEFT$(A$,39-A):
43 IFA+LEN(A$)<39THENPRINT" ";
45 POKE54273,36:POKE54272,85:POKE54277,32:POKE54278,0:POKE54296,15:POKE54276,129
46 FORZ=1TO10:NEXT:POKE54276,0
50 PRINT"(UP)":NEXT:PRINT:FORZ=1TO200:NEXT:GOTO30
110 BE=54272:POKEBE+5,85:POKEBE+6,85:POKEBE+12,85:POKEBE+13,85
115 POKEBE+24,15
120 POKEBE+4,33:POKEBE+11,17
125 FORX=1TO6:READH1,L1,H2,L2:POKEBE+1,H1:POKEBE,L1:POKEBE+8,H2:POKEBE+7,L2
130 IFH1=50THENFORT=1TO200:NEXT
135 FORT=1TO100:NEXT
150 NEXT:FORZ=54272TO54295:POKEZ,0:NEXT:GOTO270
```

Programmet fortsätter på sidan 32

Forts från sidan 31

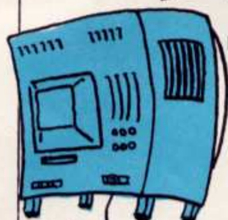
"DEMO-program"



```
155 POKE53280,0:POKE53281,0:PRINT"(CLR)"TAB(12)"(PUR)VIC PRESENTERAR"
156 PRINTTAB(12)"(CYN)———————————"
157 PRINT"(DOWN)(DOWN)(RGHT)(RGHT)(WHT)DETTA ÄR ETT DEMONSTRATIONS PROGRAM"
158 PRINT"(RGHT)(RGHT)FöR VIC-64."
159 PRINT"(DOWN)(RGHT)(RGHT)(YEL)AV CHRISTER HARRYSSON (RED)(C) 1984"
161 FORZ=1TO1000:NEXT
163 FORZ=245TO251:FORW=0TO62:READQ:POKEZ*64+W,Q:NEXT:NEXT:RETURN
164 REM ** SPRITE 1
165 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
166 DATA0,14,123,0,56,123,0,224,51,3,255,254,15,255,255,63,255,255
167 DATA255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0
168 REM ** SPRITE 2
169 DATA0,0,16,0,0,48,0,0,127,68,64,240,136,129,240,17,3,240,34,7,240
170 DATA255,255,240,255,255,240,255,255,240,255,255,243,255,255,247
171 DATA255,255,127,127,254,255,191,253,255,223,251,247,239,247,243
172 DATA15,240,0,7,224,0,3,192,0,1,128,0
173 REM ** SPRITE 3
174 DATA0,0,0,0,0,0,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
175 DATA255,128,0,255,224,0,255,255,0,255,255,240,255,255,0
176 DATA255,224,0,255,128,0,0,0,0,0,0,0,0,0,0,0
177 REM ** SPRITE 4
178 DATA0,0,0,0,0,0,255,255,255,0,62,0,0,127,0,3,127,128,7,191,240,55,223,120
179 DATA127,222,252,119,238,252,247,253,252,251,255,252,251,251,232,221,245,220
180 DATA191,239,188,191,223,188,239,255,252,111,223,252,39,223,184
181 DATA1,135,128,0,0,0
182 REM ** SPRITE 5
183 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,15,248,1,24,14,31,42,35,240
184 DATA74,40,7,138,43,132,73,74,7,40,138,53,24,10,7,15,227,128,0,56,63
185 DATA0,15,224,0,0,0,0,0,0,0,0,0,0,0
186 REM ** SPRITE 6
187 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,15,255,128,248,6,0,2,24,0,82,96,0
188 DATA82,128,0,112,96,0,18,24,0,16,6,0,7,255,128,124,0,0,192,0,0
189 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0
190 REM ** SPRITE 7
191 DATA0,127,0,1,255,192,3,255,224,3,231,224
192 DATA7,217,240,7,223,240,7,217,240,3,231,224
193 DATA3,255,224,3,255,224,2,255,160,1,127,64
194 DATA1,62,64,0,156,128,0,156,128,0,73,0,0,73,0
195 DATA0,62,0,0,62,0,0,62,0,0,28,0
250 DATA"(BLK)MANIC SOFTWARE","(BLU)PRESENTERAR","(ORNG)VIC-64","*"
253 REM ** FANFAR **
255 DATA25,30,18,209,33,135,25,30,42,62,31,165,50,60,37,162,42,62,31,165,50,60
260 DATA37,162
270 V=53248:POKEV+1,150:POKEV+3,150:POKEV+5,150:POKEV+7,134:POKEV+9,134
272 POKEV+39,6:POKEV+40,6:POKEV+41,15:POKEV+42,0:POKEV+43,0
274 POKEV,159:POKEV+2,207:POKEV+4,255:POKEV+6,47:POKEV+8,95
276 POKE2040,245:POKE2041,246:POKE2042,248:POKE2043,249:POKE2044,250
280 POKEV+23,255:POKEV+29,255:POKEV+16,24:POKEV+21,255
285 POKE54296,15:POKE54273,8:POKE54272,147:POKE54277,126:POKE54278,0
287 POKE54276,129:FORZ=1TO3000:NEXT:POKE2042,247:POKEV+41,2
290 POKE54273,36:POKE54272,85:POKE54278,16
295 FORA=0TO4:B(A)=PEEK(V+A*2):NEXT
297 FORA=0TO4:B(A)=B(A)-5:IFB(A)<0THENB(A)=0
300 NEXT
310 IFA=3OR4THEN330
320 NEXT:GOTO295
330 IFB(3)=0ANDGG=0THENPOKEV+16,PEEK(V+16)AND255-2↑3:B(3)=252
335 IFB(4)=0ANDGG=0THENPOKEV+16,PEEK(V+16)AND255-2↑4:B(4)=255:GG=1
337 IFGG=1ANDB(4)=0THEN341
340 FORA=0TO4:POKEV+A*2,B(A):NEXT:GOTO295
341 POKE54278,1:A$="3995:-":X=38
345 POKE54273,36:POKE54272,85:POKE54277,32:POKE54278,0
346 POKE54296,15:POKE54276,129
347 PRINT"(HOME)(BLU)"LEFT$(N$,22)"                (LEFT)(LEFT)(LEFT)(LEFT)(LEFT)(LEFT)(L
EFT)";
350 PRINTLEFT$(S$,X)LEFT$(A$,39-X);:X=X-1:IFX=0THENX=38:R=R+1
352 POKE54276,129:FORZ=1TO10:NEXT:POKE54276,0
356 IFR=2ANDX=17THEN365
360 PRINT"(LEFT) (UP)":GOTO347
365 POKE54287,1:POKE54286,1:POKE54291,1:POKE54292,255:POKE54296,15
370 POKE54290,33:FORZ=100TO200STEP5:POKE54286,Z:POKE54287,Z/5:NEXT
375 FORZ=200TO100STEP-5:POKE54286,Z:POKE54287,Z/5:NEXT
380 FORZ=100TO255STEP5:POKE54286,Z:POKE54287,Z/5:NEXT
390 FORZ=1TO3000:NEXT:POKE53280,5:POKEV+21,0
400 PRINT"(CLR)(DOWN)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT
)(RGHT)(RGHT)(RGHT)(RGHT)(BLK)TEKNISK DATA":PRINTTAB(14)"———————————"
410 PRINT"(DOWN)(RGHT)(RGHT)(RED)64K RAM (39K FöR BASIC) . (PUR)20K ROM"
415 PRINT"(DOWN)(RGHT)(RGHT)(BLU)40 X 25 RUTOR PA SKÄRMEN"
420 PRINT"(DOWN)(RGHT)(RGHT)(GRN)8 SPRITES PA 24 X 21 PUNKTER"
425 PRINT"(DOWN)(RGHT)(RGHT)(BRN)HöGUPPLöSNING 320 X 200 PUNKTER"
430 PRINT"(DOWN)(RGHT)(RGHT)(ORNG)LJUD:":PRINT"(DOWN)(RGHT)(RGHT)(RED)3 STÄMMOR
MED VARDERA 9 OKTAVER"
435 PRINT"(DOWN)(RGHT)(RGHT)(GRY2)4 VAGFORMER,FILTER OCH VOLYMKONTROLL"
440 PRINT"(DOWN)(DOWN)(RGHT)(RGHT)(BLK)66 TANGENTER (INKL. AÄö)"
445 PRINT"(DOWN)(RGHT)(RGHT)(BLU)STORA OCH SMA BOKSTÄVER,GRAFIK TECKEN"
450 PRINT"(DOWN)(RGHT)(RGHT)(RED)PRIS: (BLU)3995:- (RED)(INKL. AÄö OCH MANUAL)"
460 L1=54272:L2=54279:L3=54286
470 H1=L1+1:H2=L2+1:H3=L3+1
480 V1=L1+4:V2=L2+4:V3=L3+4
490 POKE54296,15:Z=0
500 POKEV1+1,9:POKEV1+2,0
510 POKEV2+1,36:POKEV2+2,36
520 POKEV3+1,18:POKEV3+2,250
530 T=TI
540 POKEV1,16:POKEV2,32:POKEV3,16
550 READS:IFS=0THEN630
560 READX1,Y1,X2,Y2,X3,Y3
570 IFX1THENPOKEH1,X1:POKEL1,Y1:POKEV1,17
580 IFX2THENPOKEH2,X2:POKEL2,Y2:POKEV2,33
590 IFX3THENPOKEH3,X3:POKEL3,Y3:POKEV3,17
600 T=T+S
610 IFT>TITHEN610
620 GOTO540
630 FORJ=L1TO54296:POKEJ,0:NEXT
640 DATA10,51,97,0,0,0,0,10,43,52,0,0,0,0
641 DATA20,34,75,21,154,8,147,20,34,75,25,177,0,0
642 DATA10,34,75,17,37,6,108,10,38,126,0,0,0,0
```

Programmet fortsätter på sidan 34

SPELREKORD



Nu har vi satt de två spelen med 'VIC-1901' som nummer under samma rubrik och därför kommer 'AVENGER' nu på plats i VIC 20 spalten. I 64 spalten har 'LE MANS' bytts ut mot det mer 'heta' 'ZAXXON'.

Ni får gärna kopiera eller skriva av 'rutan', så slipper ni klippa sönder VIC rapport.

SPELREDAKTÖREN C.E.J.



JELLY MONSTERS VIC-1905

1. 2 601 160 Peter Johansson, Lyckeby
2. 1 631 860 Pelle Gustavsson, Sundbyberg
3. 705 980 Anders Hellman, Nol

RAT RACE VIC-1909

1. 101 520 Pär-Olof Håkansson, Bjärred
2. 96 200 Olli Pesonen, Angered
3. 92 000 Jocke Blyborg, Strängnäs

JUPITER & SUPERLANDER VIC-1907

1. 139 900 Björn Lindman, Huddinge
2. 120 200 Paul Jönsson, Genarp
3. 108 200 Dan Andersson, Södra Sandby

OMEGA RACE VIC-1924

(5 skepp)

1. 306 900 Fredrik Eklund, Linköping
2. 296 400 Tomas Gordström, Linköping
3. 256 160 Anders Togerö, Varberg

AVENGER VIC-1901

1. 5 800 Thomas Johansson, Ystad
2. 4 340 Pelle Gustavsson, Sundbyberg
3. 3 560 Roger Ulvnäs, Uppsala



CUP-FINAL (Commodore)

(Svårighetsgrad: 9)

1. 14–0 Peder Isaksson, Ösmo
2. 8–0 Ola Stjärnhagen, Höör
3. 8–1 Fredrik Nordblad, Tärondö

GRIDRUNNER (Llamasoft)

(Level 1–)

1. 672 529 Morgan Bergquist, Matfors
2. 480 720 Thomas Hartman, Luleå
3. 461 700 Jörgen Forsberg, Märsta

JUMPMAN (EPYX)

(Grand loop)

1. 61 500 Sverker Edbladh, Landskrona
2. 58 800 Peter Blomgren, Handen
3. 53 025 Mattias Lindh, Skellefteå

JUMPMAN JR (EPYX)

(Valfri hastighet)

1. 329 250 Mats Wester, Täby
2. 213 360 Anders Olsson, Gråbo
3. 213 025 Thomas Ljunggren, Kungsbacka

ZAXXON (SEGA)

(3 skepp)

1. 111 350 Glenn Johansson, Stenungsund
2. 101 700 Magnus Wennström, Stenungsund
3. 45 150 Magnus Persson, Kristianstad



VIC 20 ☐ VIC 64 ☐ Datum

Namn ....

Adress ....

Postnr .... Postadress ....

Spel ....

Erhållna poäng ....

Intygas av målsman eller annan myndig person.

Forts från sidan 32

"DEMO-program"

```
643 DATA10,43,52,25,177,0,0,10,45,198,0,0,0,0
644 DATA20,51,97,21,154,8,147,20,51,97,25,177,0,0
645 DATA20,51,97,17,37,9,159,20,43,52,25,177,10,205
646 DATA20,57,172,22,227,11,114,20,57,172,34,75,0,0
647 DATA20,57,172,28,214,8,147,10,0,0,34,75,0,0
648 DATA10,51,97,0,0,0,0,20,57,172,22,227,11,114
649 DATA10,0,0,34,75,0,0,10,51,97,0,0,0,0
650 DATA10,57,172,28,214,10,205,10,64,188,0,0,0,0
651 DATA10,68,149,34,75,9,159,10,76,252,0,0,0,0
652 DATA10,76,252,0,0,0,0,20,86,105,21,154,8,147
653 DATA20,0,0,25,177,0,0,20,0,0,17,37,6,108
654 DATA10,68,149,25,177,0,0,10,51,97,0,0,0,0
655 DATA20,68,149,21,154,8,147,20,0,0,25,177,0,0
656 DATA20,0,0,17,37,8,23,10,51,97,25,177,7,53
657 DATA10,43,52,0,0,0,0,20,51,97,22,227,6,108
658 DATA20,0,0,25,177,0,0,20,0,0,19,63,9,159
659 DATA10,38,126,25,177,0,0,10,43,100,0,0,0,0
660 DATA20,34,75,21,154,8,147,20,0,0,25,177,6,108
661 DATA20,0,0,21,154,4,73,20,0,0,0,0,0,0
662 FORZ=1TO10000:NEXT:POKE53281,0
665 PRINT"(CLR)"TAB(10)"(WHT)VIC-64 HAR 16 FÄRGER":CD$="(BLK)(WHT)(RED)(CYN)(PUR
)(GRN)(BLU)(YEL)(ORNG)(BRN)(LRED)(GRY1)(GRY2)(LGRN)(LBLU)(GRY3)"
670 DATA"(RGHT)(BLK)SVART(RGHT)","(RGHT)(RGHT)(WHT)VIT(RGHT)(RGHT)(RGHT)","(RGHT
)(RGHT)(RED)RöD(RGHT)(RGHT)(RGHT)","(RGHT)(RGHT)(CYN)CYAN(RGHT)(RGHT)","(RGHT)(P
UR)PURPUR(RGHT)"
672 DATA"(RGHT)(RGHT)(GRN)GRöN(RGHT)(RGHT)","(RGHT)(RGHT)(BLU)BLA(RGHT)(RGHT)(RG
HT)","(RGHT)(RGHT)(YEL)GUL(RGHT)(RGHT)(RGHT)","(RGHT)(ORNG)ORANGE(RGHT)","(RGHT)
(RGHT)(BRN)BRUN(RGHT)(RGHT)"
674 DATA"(RGHT)(LRED)LJ RöD(RGHT)","(RGHT)(GRY1)GRA 1(RGHT)(RGHT)","(RGHT)(GRY2)
GRA 2(RGHT)(RGHT)","(LGRN)LJ GRöN(RGHT)"
676 DATA"(RGHT)(LBLU)LJ BLA(RGHT)","(RGHT)(GRY3)GRA 3(RGHT)(RGHT)"
680 FORX=0TO3:FORY=1TO5:FORZ=1TO4
685 PRINT" (RVS)"MID$(CD$,X*4+Z,1)"          (OFF) ";
690 NEXTZ:NEXTY:IFX<>3THENPRINT
695 NEXTX:PRINT"(HOME)(DOWN)(WHT)┌────────┐"SPC(30)"|        |"
700 PRINT"|        |"SPC(30)"|        |"
705 PRINT"└────────┘"
710 PRINT"(HOME)";:FORX=1TO4:READA$,B$,C$,D$
715 PRINT"(DOWN)(DOWN)(DOWN)(DOWN)";:IFX=1THENPRINT"(UP)";
720 PRINT"(RGHT)(RVS)"A$,"(RGHT)(RVS)"B$,"(RGHT)(RVS)"C$,"(RGHT)(RVS)"D$:NEXT
723 FORZ=1TO5000:NEXT
725 FORX=0TO16:FORZ=1TO1500:NEXT:POKE53281,XAND15:NEXT:FORZ=1TO6000:NEXT
730 POKE53280,7:POKE53281,1:PRINT"(CLR)(DOWN)(RGHT)(RGHT)(BRN)VIC-64 KAN HÄRMA O
LIKA SAKER"
732 PRINT"(RGHT)(RGHT)(DOWN)HöR HÄR:(BLU)":BE=54286:POKE54296,15:FORZ=1TO1000:NE
XT
735 POKEBE,85:POKEBE+1,36:POKEBE+2,255:POKEBE+3,0
740 PRINT"(DOWN)(RGHT)(RGHT)GITARR":POKEBE+5,9:POKEBE+6,0:FORO=1TO4:POKEBE+4,65
745 FORZ=1TO800:NEXT:POKEBE+4,0:NEXT:FORZ=1TO500:NEXT
750 PRINT"(DOWN)(RGHT)(RGHT)FLöJT":POKEBE+5,96:POKEBE+6,240:POKEBE+4,17:FORZ=1TO
1500:NEXT
755 POKEBE+4,0:FORZ=1TO500:NEXT
760 PRINT"(DOWN)(RGHT)(RGHT)XYLOFON":POKEBE+5,9:POKEBE+6,0:FORO=1TO4:POKEBE+4,17
765 FORZ=1TO800:NEXT:POKEBE+4,0:NEXT:FORZ=1TO500:NEXT
770 PRINT"(DOWN)(RGHT)(RGHT)ORGEL":POKEBE+5,0:POKEBE+6,240:POKEBE+4,17
775 FORZ=1TO1500:NEXT:POKEBE+4,0:FORZ=1TO500:NEXT
780 PRINT"(DOWN)(RGHT)(RGHT)TRUMPET":POKEBE+5,96:POKEBE+6,240:POKEBE+4,33
785 FORZ=1TO1500:NEXT:POKEBE+4,0:FORZ=1TO500:NEXT
790 PRINT"(DOWN)(RGHT)(RGHT)HAVETS BRUS":POKEBE+5,236:POKEBE+6,0:POKEBE+4,129
795 FORZ=1TO8000:NEXT:POKEBE+4,0:FORZ=1TO1000:NEXT
800 POKE53280,4:V=53248:PRINT"(CLR)(DOWN)(DOWN)"TAB(10)"(BLK)SPRITES PA VIC-64"
803 A$="  EN SPRITE KAN HA 4 OLIKA STORLEKAR."
805 FORA=2040TO2043:POKEA,251:NEXT:FORA=V+39TOV+42:POKEA,2:NEXT
810 FORA=V+1TOV+7STEP2:POKEA,100:NEXT
815 POKEV,80:POKEV+2,128:POKEV+4,176:POKEV+6,224
820 POKEV+23,10:POKEV+29,12:POKEV+21,15
825 FORZ=1TO37
830 PRINT"(HOME)(BLU)"LEFT$(N$,15)LEFT$(A$,Z):FORQ=1TO60:NEXT:NEXT
835 FORZ=1TO6000:NEXT:PRINT"(CLR)"
840 POKEV,0:POKEV+2,0:POKEV+21,3:POKEV+23,2:POKEV+29,2:POKEV+40,6
845 PRINT"(HOME)(DOWN)(RGHT)(RED)SPRITES KAN GA BAKOM VARANDRA"
850 FORZ=0TO255:POKEV,Z:POKEV+2,255-Z:FORQ=1TO50:NEXT:NEXT
855 POKEV+21,0:POKEV,0:POKEV+3,140:GOTO865
860 PRINT"(CLR)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)(DOWN)":FORA=1TO10:PRINTTAB(1
3)"(BLK)(RVS)      (OFF)(RED)ABCDEF(RVS)(BLK)      (OFF)":NEXT:RETURN
865 GOSUB860:PRINT"(HOME)(DOWN)(RGHT)SPRITES KAN GA FRAMFöR TEXTEN.":POKEV+27,0:
GOSUB880
870 POKEV,0:PRINT"(HOME)(DOWN)(DOWN)(DOWN)(RGHT)ELLER BAKOM TEXTEN.":POKEV+27,2
875 GOSUB880:POKEV+27,0:GOTO890
880 POKEV+21,2:FORA=0TO255:POKEV+2,A:FORZ=1TO50:NEXT:NEXT:RETURN
890 POKEV+21,0:PRINT"(CLR)":POKE53280,5
895 PRINT"(HOME)(DOWN)(DOWN)(RGHT)(RGHT)(RGHT)(RGHT)(RED)VIC-64 HAR 255 OLIKA TE
CKEN"
900 FORA=0TO255:POKE1224+A*2,A:POKE1224+A*2+54272,6:NEXT
905 FORZ=1TO10000:NEXT
910 PRINT"(CLR)(DOWN)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(RGHT)(BLK)HöGUPPLöSNIN
G PA VIC-64"
915 PRINT"(DOWN)(DOWN)(RGHT)(BLU)SKÄRMEN BESTAR DA AV 320 X 200 PUNKTER."
920 PRINT"(DOWN)(RGHT)FöRST PLOTTAS EN CIRKEL, OCH SEDAN":PRINT"(RGHT)EN SINUS-K
URVA"
925 BM=16384:FM=24576
930 FORI=BMTOBM+8000:POKEI,0:NEXT:FORI=FMTOFM+1000:POKEI,224:NEXT
935 POKE53281,0:POKE56576,PEEK(56576)AND252OR2
940 POKE53265,PEEK(53265)OR32:POKE53272,128
945 FORA=0TO2*πSTEP.04:X=INT(COS(A)*80+159)
950 Y=INT(SIN(A)*80+99):GOSUB1000:NEXT
955 FORX=0TO319:Y=99+INT(SIN(X/25)*90)
960 GOSUB1000:NEXT
970 FORZ=1TO1000:NEXT:POKE53265,PEEK(53265)AND223
975 POKE56576,151:POKE53272,21
980 RUN
1000 REM ** PLOT **
1010 ADR=BM+8*INT(X/8)+320*INT((199-Y)/8)+((199-Y)AND7)
1020 POKEADR,PEEK(ADR)OR2↑(7-(XAND7))
1030 RETURN

READY.
```

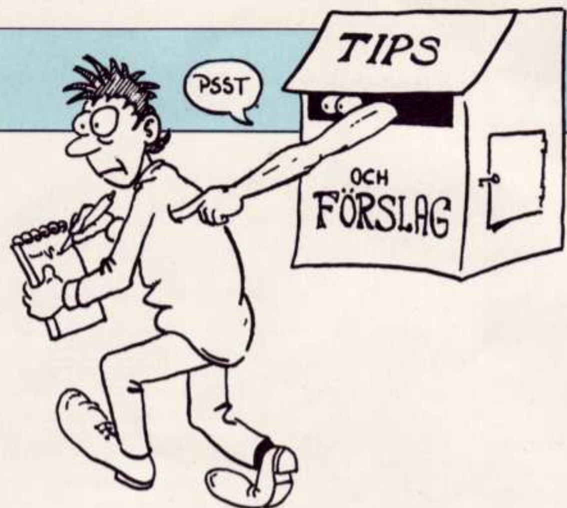
I nästa nummer av VIC rapport presenterar vi 2-pristagarna

Program till artikel "Kopiera skärmen..."

```
100 REM**FAM AUG 1984**
110 FOR2:FOR0:A$="ABCDEFG":PZ$="":FORI=1TO7:PZ$=PZ$+A$:NEXT
120 I=VAL(PZ$):POKE251,PEEK(71):POKE252,PEEK(72):I=PEEK(251)+256*PEEK(252)
130 SY=PEEK(I+1)+256*PEEK(I+2)
140 FORI=SYTOSY+48:READX:POKEI,X:NEXT
150 DATA 169 , 80 , 133 , 69 , 169 , 210 , 133 , 70
160 DATA 32 , 231 , 208 , 160 , 1 , 177 , 71 , 133
170 DATA 253 , 200 , 177 , 71 , 133 , 254 , 162 , 7
180 DATA 138 , 168 , 177 , 251 , 133 , 255 , 160 , 7
190 DATA 102 , 255 , 8 , 177 , 253 , 106 , 40 , 234
200 DATA 145 , 253 , 136 , 16 , 243 , 202 , 16 , 232
210 DATA 96
220 REM
230 REM**EXEMPEL PÅ HöGUPPLöSNINGSGRAFIK MED SUPER EXPANDER**
240 FOR2:GOTO19,0,"FAM DESIGN":A=480:READ2,A+A,A
250 FORI=2.8TO157.8TEP2.8:INPUT#2TOA+A*COS(I),A+A*SIN(I):NEXT
260 GOSUB290:FOR0:END
270 REM
280 REM**SUBRUTIN**
290 CM=PEEK(36869)AND8:CC=0
300 IFCM=0THENCC=32768
310 CM=CC+(PEEK(36869)AND7)*1024
320 SM=(PEEK(36869)AND112)*64+(PEEK(36866)AND128)*4
330 NH=(PEEK(36867)AND1)+1
340 NR=(PEEK(36867)AND126)/2
350 NC=PEEK(36866)AND127
360 N$="":FORI=1TO8:N$=N$+CHR$(0):NEXT:PR$=N$
370 PM$="       ":PL=251:PH=252:CC=8*NH:C8=256
380 OPEN 3,4,3:PRINT#3,CHR$(114):CLOSE3
390 OPEN5,4,5
400 OPEN6,4,6
410 OPEN4,4:PRINT#4,CHR$(147)
420 FORI=0TONR-1:SK=SM+I*NC
430 FORK=0TO8*NH-1STEP8:PRINT#6,CHR$(0):PP$=PM$
440 FORJ=0TONC-1:CA=CM+CC*PEEK(SK+J)+K:CH=INT(CA/C8)
450 POKEPL,CA-C8*CH:POKEPH,CH:SYSSY:IFPR$=N$THEN470
460 PRINT#5,PR$:PRINT#4,PP$+CHR$(254)
470 PP$=PP$+" "
480 NEXTJ:PRINT#6,CHR$(21):PRINT#4
490 NEXTK:NEXTI
500 CLOSE3:CLOSE4:CLOSE5:CLOSE6:RETURN

READY.
```

TIPS & FÖRSLAG



Många funderingar

Jag köpte mig ett nummer av VIC rapport (nr 5/6) och började läsa. När jag kom till sidan 80 blev jag fundersam över det där om att läsa av det man skrivit på skärmen. Det är ju klart att om man är sådan att man tycker att POKE 780,65:SYS 65490 är bättre än PRINT "A" så gärna för mig. Men det där programmet hade jag skrivit så här:

```
10 PRINT CHR$(147):OPEN 3,3
20 PRINT "VAD DU VILL"+CHR$(145)
30 INPUT #3,B$
40 PRINT B$
```

Rad 10 öppnar en fil till bildskärmen.

Sedan har jag ett tips för dig som skriver korta maskinkodsrutiner. Området mellan $033C och $3FF är ju populärt att rekommendera, men använder man bandspelaren så är allt förstört, så använd $02A1 till $02FF där finns det inget som stör. Observera att området är något mindre på VIC 64.

Ett annat tips jag vill varna för är att gå in och ändra i CHRGET-rutinen om man vill göra egna kommandon. Man ska helt enkelt ha tur för att få det att fungera *helt* felfritt. Jag har den senaste tiden gjort egna kommandon och måste säga att det klart enklaste är att gå in och ändra på BASICvektorerna. Precis hur man gör är för komplicerat för att beskriva i några få rader men man kan studera de rutiner som ligger i operativsystemet.

```
10 FORT=688to734:READA:POKET,A:NEXT:SYS724
20 DATA 32,115,0,201,64,240,6,32,121,0
30 DATA 76,231,199,32,115,0,32,158,215,138
40 DATA 72,32,253,206,32,158,215,104,168,24,32,240,
   255
50 DATA 76,174,199,169,2,162,176,142,8,3,141
   ,9,3,96
```

Det här programmet gör att "@" "AT" blir ett kommando som styr markören. T ex @10,9 flyttar markören till rad nio, tio steg ut. För att man ska kunna använda programmet till VIC 64 måste man ändra 199 på rad 30 och 50 till 167, 206 på rad 40 till 174, och 215 på rad 30 och 40 till 183. Det här programmet går inte att använda tillsammans med en programmeringshjälp typ SUPER EXPANDER eller SIMON'S BASIC. □

Johan Lagerros

Calc Result Easy

Christer Svensson kommer med följande lilla tips till alla som använder Calc Result Easy.

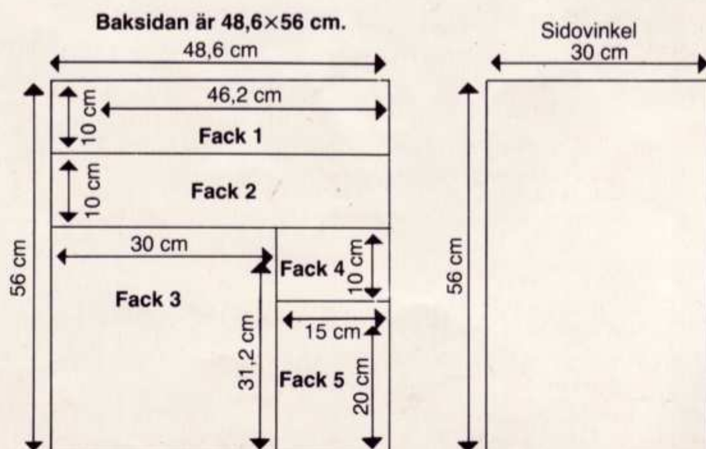
Normalt när man gör nya beräkningar på Calc Result så lagras dessa vid sidan om de gamla beräkningarna. Till slut sitter man med ett stort bibliotek av beräkningar. Men om man trycker på
@0:gamla filnamnet
så ersätter de nya beräkningarna de gamla. Alltså behöver man bara lagra aktuella beräkningar på disketten.

DATORSKÅP

Jag har ett förslag på ett datorskåp där du kan förvara din dator + tillbehör. Du behöver en såg, spånplatta eller annat brett träslag, tumstock, en hyvel, trälim och skruvar. Det är till listen du behöver trälim. (Använd 12 mm spånskiva.)

Användning av facken:
1. Dator VIC 20 eller CBM 64
2. Kassettbandspelare, joystick och band
3. Tidningar och böcker
4. Disketter eller cartridge
5. Sladdar och generator

Niklas Lundin



Auto start VIC 20

Du har kanske undrat hur vissa program startar automatiskt fastän du skrivit "load". Jag ska här beskriva hur det fungerar: Man låter adresserna $0302–3(770–771) peka mot din egen maskinkod som är placerad mellan $02A7–$02FF. Maskinkoden är sedan save:ad. Efter ett "load" hoppar maskinen alltid dit adresserna 770–771 pekar. Normalt pekar de mot en rutin som bl a skriver ut "Ready" på skärmen.

Maskinkoden läggs in med det här programmet:

```
10 FOR L = 745 to 767 : READ
A:POKE L,A:NEXT
100 DATA 169, 131, 141, 2, 3, 169,
196, 141, 3, 3, 169, 131, 141, 119,
2, 169, 1, 141, 198, 0, 76, 131, 196
200 NEW
```

Maskinprogrammet ställer tillbaka adresserna till sina normala värden. Den lägger också in 131 i tangentbordsbufferten, sedan hoppar den till rutinen som skriver ut "Ready".

Därefter ändrar man pekarna till "start of Basic" och "Pointer to end of program" med följande pokesatser.

```
POKE 43,233:POKE 44,2:POKE
45,4:POKE 46,3
```

ändra sedan 770–1 till början av maskinkoden.

```
POKE 768,34:POKE 769,253:
POKE 770,233:POKE 771,2:SA-
VE "......",1,3
```

När det kommer upp "Ready" och under det "Load searching" tryck då på stoppknappen. Loada nu in programmet du vill ska Autostarta. Spela tillbaka bandet med Auto-starten. Skriv "Verify" och tryck på play. När det kommer upp "veryfyingerror" save:ar du ditt eget program på vanligt sätt. Spela tillbaka bandet och tryck in "Load" och se, ditt program startar automatiskt. □

Fredrik Ejhed

C 2 N-bandspelare

Jag har mot alla varningar öppnat min C 2 N-bandspelare. Jag blev förvånad över vad jag såg. Där fanns ett kretskort och en utskjutande kontakt som passade kontakten på kabeln. Nu undrar jag varför?

Jag undrar också om det är någon skillnad mellan C 2 N och Supersaver 20/64.

Johan Rofors

Svar:

Jag testade kontakten du nämnde för att se om man kan använda den vid kopiering eller liknande. Detta fungerade inte då datorn inte tillför tillräckligt med ström. Troligen är kontakten en testkontakt som används för att kontrollera att bandspelaren fungerar korrekt innan man levererar den.

Angående din andra fråga om skillnaden mellan C 2 N och Supersaver 20/64 kan jag inte ge det ett mer utförligt svar än: Mig veterligen är det ingen större skillnad mellan de två.

Supersaver 20/64 är något billigare vilket kan avspegla sig i kvalitén. Men annars tror jag inte att det är någon större skillnad.

Fel funktion

I boken "Lär dig VIC 20" och programmeringshandboken till VIC 20 står det att för hyperbolic tangent i BASIC är:

TANH(x)=EXP(−x)/EXP(x)
+EXP(−x)*2+1

Om x=0,5 blir svaret 2,58094076. På min miniräknare, Casio fx-3600 p, är hyp. tan. av 0,5 = 0,462117157. Med min formel

EXP(−x)/(EXP(x)
+EXP(−x))*−2+1

blir svaret detsamma som på miniräknaren. Nu undrar jag om det är formeln i böckerna eller min miniräknare som det är fel på? Om formeln är fel, är då även formeln för hyperbolic cotangent fel?

COTH(x)=EXP(−x)/(EXP(x)
−EXP(−x))*2+1

Stefan Spjuth

Svar: Den första formeln är fel som du mycket riktigt påpekar. Däremot är den andra formeln rätt om man bara ser till att parentesen hamnar rätt kring uttrycket (EXP(x)−EXP(−x)). □

Kjolavipsen

(eller kanske kjolavicken chipsen)

Jag heter Jimmy Engelbrekt och är prenumerant på VIC rapport vilken jag tycker är bra. Jag har en VIC 20 och en del idéer. Först undrar jag om VIC rapport inte kan göra några pärmar med VIC motiv på i vilka man kan spara sina VIC rapport.

För det andra tror jag att det är många VIC ägare som tänkt på att man inte kan få ut citationstecken på skärmen på vanligt sätt. Jag har kommit på att man kan genomföra det med PRINTCHR$(34).

Svar:

Jimmy, tack för dina förslag. Vi ska överväga att göra en samlingspärm för VIC rapport. Vi återkommer om detta i senare nummer.

Red.

Maskinkodsprogrammering

Var eller hur kan jag lära mig programmera i maskinkod? Hur gör man egna tecken i högupplösning?

Frågande

FRÅGOR & SVAR

Svar:

För den som vill lära sig programmera i maskinkod finns en mängd litteratur på marknaden. Vad du ska leta efter är böcker om 6502-assembler. Assembler och maskinkod är nämligen två benämningar för samma sak. Den som håller på med maskinspråk skriver dock direkt de koder som processorn föreslår medan assemblerprogrammerarna använder ett program som översätter ett symboliserat språk till just maskinkod.

Ett gott råd är att följa den assemblerserie som du hittar i VIC rapport.

Åke Hedman

Finns modem för VIC 20?

Hej igen! Jag tänker fråga om modem. Det är dåligt med modem för VIC 20.

Finns det överhuvud taget ett modem för VIC 20 med terminalprogram.

Lauri Lundgren

Svaret finns på nästa sida

Forts från sidan 37

Svar:

Modem är samma som till 64an, dvs Handics 75/1200 & 300/300 modem.

Programvaran är det däremot sämre med. Det finns gott om sådant till 64an, men mig veterligen inget till VIC 20, för 300/300 baud kan man skriva programmet själv i Basic, men för 1200 baud krävs maskinspråk. □

Måns

Svåra ord

Jag är en ny tidningsköpare VIC rapport och i varje artikel (nästan) stöter jag på samma problem. De ord som används förstår jag inte och därför får artikeln inget värde, eftersom missförstånd lätt blir följden om man ska försöka tolka orden. Eftersom många ord har sin speciella yrkeskaraktär i datasvängar finns många gånger inte den rätta definitionen i en vanlig ordlista. Så där sitter man med en tidning som man inte kan läsa med full behållning. Kunde VIC rapport därför inte starta med en ordlista i varje tidning?

Då jag är nybörjare har jag haft problem att lära mig att använda VIC 64. Problemet ligger bl a i läromedlets uppbyggnad. Gradienten för kunskapsförmedlingen är för brant i de flesta böcker. En bok börjar med att beskriva hur man använder band och flexskiveenheter innan den knappt beskrivit vad ett Basic-kommando är för något. Om en bok är väldigt virrigt skriven med olika kunskapsnivåer blandade hur som helst i texten, tar det väldigt lång tid att hoppa fram och tillbaka i boken för att försöka tolka vad som står skrivet i den aktuella texten. Samma fel kan man hitta i nästan vilken teknisk text som helst. Slå ett slag för bättre läromedel!

Åke Södergren

Svar:

VIC rapport hade redan i våras tänkt starta en avdelning i vilken de svåraste orden togs upp. Problemet är att man blir hemmablind och får svårt att avgöra vad som är självklart och vad som verkligen är svårt. Kanske vi gör ett försök igen i senare nummer av VIC rapport.

Vi slår gärna ett slag för bättre läromedel. Vi kan hålla med om att mycket datalitteratur inte håller den standard man kan önska. Vi får hoppas att det sker en uppryckning i framtiden med tanke på datorns utveckling och spridning. □

Återkommande artikelserier

Jag brukar köpa VIC rapport som lösnummer och tycker att den är väldigt bra. Men det är ganska irriterande att vissa artikelserier t ex Assemblerskolan inte är med i varje nummer. Det verkar som om artiklarna slängs in spontant. Dessutom fattas en del programlistningar.

Anders Holmberg

Svar:

VIC rapports artikelserier är avsedda att publiceras i varje nummer fram tills dess att de avslutas. Våra artiklar är normalt mycket planerade och slängs inte in slumpmässigt.

Klagomål på tryckfel och utelämnade programlistningar är helt berättigade. Vi ska försöka undvika dessa fel i framtiden bl a med hjälp av längre pressläggningstid. Så om det tar lång tid innan ni hittar era bidrag i tidningen så beror det på detta. □

Red.

Spökrader och bomber

Jag har två frågor jag vill få svar på:

1. Vid LOAD och SAVE kan man använda en andra siffra efter device-adressen. Vad har den för funktion i direkt-mode och i program-mode? Jag har tittat i bland annat Programer's Reference Guide utan att kunna hitta någon enkel och utförlig förklaring.

2. I ett program jag jobbar med har jag flyttat BASIC-starten till $4000, men jag har fått vissa problem med att ändra i programmet, det dyker upp "spökrader" i slutet av programmet och ibland "bombar" datorn när jag försöker ändra. Jag har HES MON 64. Vad beror felen på?

Joakim Malmén

Svar:

1. Den andra siffran efter LOAD och SAVE kallas för secondary address och används för att skicka speciella kommandon till floppy och bandspelare. De användbara formerna av secondary address är som följer

LOAD"NAMN",8,1

Den sista ettan i detta fall innebär att programmet kommer att laddas in till det ställe där det sparades ifrån, d v s inte nödvändigtvis till BASICstarten $0800.

SAVE"NAME"1,2

Tvåan innebär att programmet kommer att sparas på band med en EOT-markering sist. Detta faktum kommer att märkas på så vis att bandspelaren kommer att stanna med felmeddelandet "FILE NOT FOUND" om rätt program inte har hittats. EOT betyder nämligen End Of Tape och ger en signal att bandet har tagit slut, som namnet antyder.

I princip skulle man kunna skriva vilket nummer som helst mellan 0 och 255 efter device-numret, men så vitt jag har testat ger andra värden än 1 och 2 ingen särskild effekt.

2. Ditt problem kan bero på flera saker, men så här flyttas BASIC-starten korrekt:

1. POKE 43,(START AND 255)+1
2. POKE 44, START/256
3. POKE START,0
4. NEW

Nu har alla pekare flyttats till rätt ställe, men fortfarande kan t ex POKES och RS-232 buffrar störa ditt program. Att du har en monitor kan också innebära problem, om du blandar BASIC och maskinspråk, eftersom monitorn kan använda adresser och pekare som måste lämnas intakta för att BASICEN ska fungera. □

Mats Wvk

FAMILJEN VIC TORSSON

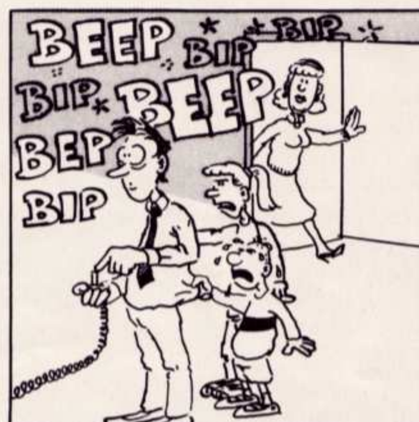




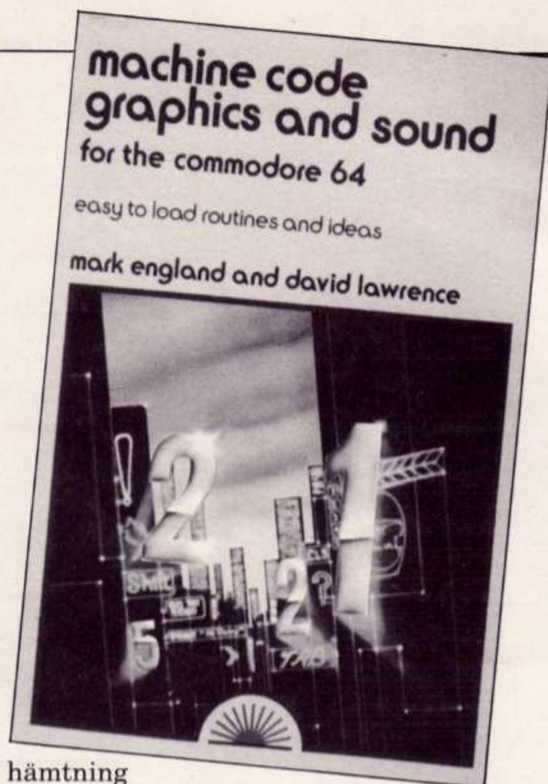








För den som älskar maskinkodning



Machinecode graphic and sound for the Commodore 64 av Mark England och David Lawrence, SUNSHINE BOOKS.

Den här boken är mer för dig som älskar maskinkodning än för dem som behöver extra kommandon för grafik och ljud. Detta beror på att samtliga rutiner måste skrivas in manuellt. Då det rör sig om ca 4000 bytes maskinkod lär det ta rätt många timmar. Till boken borde egentligen följa ett kassettband med alla program. Ett bättre alternativ för dem som vill ha grafik och ljudkommandon är att köpa t ex SIMONS BASIC. Där finns ju också en hel del annat.

Boken kan ses som en fortsättning på den tidigare utgivna "Commodore 64 machinecode master". Ett glädjande besked är att samtliga rutiner m m i den boken finns att tillgå på ett kassettband, som Sunshine Books säljer, åtminstone i England.

Har du dessa rutiner så kan du med assembler läsa in även programmen i denna bok. Detta är dock inget bra sätt, för någon kontrollsumma erhålls ej. Använd istället den i appendix givna laddrutinen av maskinkoden (med kontrollsummor). Det är naturligtvis ett mycket trist arbete att skriva in allt för hand, kanske man vid försäljning av boken i Sverige bifogar ett kassettband? Varför göra om ett arbete som författarna redan gjort. För att kommandona sen skall vara lättanvända så borde en sammanfattande användarbeskrivning ha funnits i boken.

Nu till innehållet, boken är mycket ambitiös och förklarar detaljerat hur allting går till. Nästan alla rutiner finns även i Basic för teständamål av principen. Inledningsvis finns några maskinkodsrutiner för hämtning av ord, byte och strängar från Basic-text.

LÅGUPPLÖSNINGSGRAFIK

Naturligtvis behövs inte högupplösningsgrafik alltid. Här visas hur några i teckengeneratorn redan befintliga plot-tecken används. Dessa tecken är kombinationer av fyrkanter som fyller ut fjärdedel av ett tecken t ex ◘, ◙. Dessa tecken är ju ypperliga att använda på en printer som ger ca 160*240 punkter per A4-sida, nästan som högupplösningsgrafik på VIC 64-skärmen.

De kommandon som används är BANK för angivande av videobank – LOCATE, placera cursorn – SCREEN, ange skärmminnets position – COLOUR, sätt skärmfärg och PLOT för att plotta en punkt.

HÖGUPPLÖSNINGSGRAFIK

Till att börja med förklaras hur högupplösningsgrafik fungerar. När detta är klart följer en mängd kommandon, ALLOT, reserverar minne – CCOPY, copierar alla tecken – CHRPTR, talar om för VIC chip var alla tecken finns – LOW, TURNOF, HIGH, ändrar skärmen – HSCREEN, definierar skärmparametrar – HCLEAR, rensar skärmen – HLOCATE, placerar cursorn – HPLOT, plottar – INTEGER, WINDOW, LINE PLOTTING, för linjedragning – CIRCLE, ritar ellips – TEXT, PUTCHR, skriver text på högupplösningsskärmen.

Tyvärr saknas kommandon för flerfärgsmod.

SPRITES

Efter att ha förklarat lite om sprites visas hur man sätter på (SPRITE) och stänger av (DESPRITE) en viss sprite. Färgen sätts med COLOUR. ENLARGE

VIC 20
Games 'n more

av Earl R. Savage, 158 ss ill. ISBN: 0-672-22088-1. Distr. PAGINA Förlags AB, Sthlm. Pris: 104:–.

I den flod av oöversatta böcker våra datafreaks varje år drabbas av är en del bra, andra mindre bra.

Denna bok tycker jag är bra. Anledningarna är två: programmen är korta men med möjlighet till variation, dels följer det med flödesscheman, vilka förklarar och åskådliggör programmen.

Boken är uppdelad i tre avdelningar. Den första beskriver ett antal spel, vilka till största delen utgörs av familjespel. VIC 20 får alltså spela rollen av underhållare för en hel familj. De allra flesta spelen är bekanta för oss och kan direkt tillämpas. Några är väl främmande och skulle egentligen



RECENSIONER

förstorar en sprite – SPTR medger byte av sprite med nr visst genom att hämta värdena från annan del av minnet. RESPRITE och SHAPE är bra rutiner för att skapa en sprite. Tyvärr saknas även här flerfärgsmod.

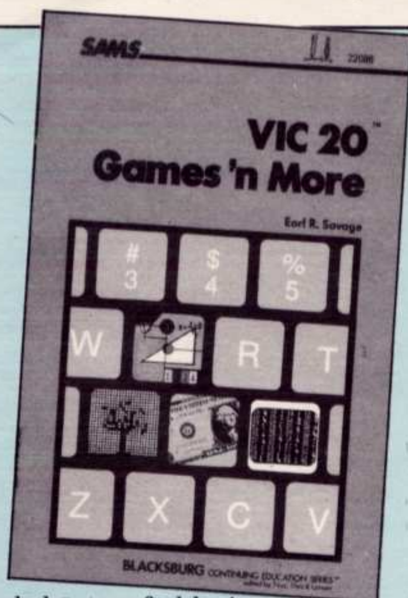
Inget nämns heller om prioritering d v s om en sprite skall röra sig bakom eller framför text eller annan sprite.

LJUD

En enkel ton sätts på med BEEP, tre stycken med CHORD. VOLUME och ADSR ställer in varje tongenerator – FILTER väljer filterparametrar – VOICE väljer vågform, koppling m m – PULSE sätter pulsbredd. Till sist det avancerade kommandot PLAY som gör det möjligt att spela in en följd av toner definierade i en sträng.

Vid en jämförelse med SIMONS BASIC är, vad gäller ljud och grafik, antalet kommandon i boken färre. Vad boken ger är en förståelse för hur maskinkodsrutiner för främst grafik och ljud genereras. När du kan detta är det bara att fortsätta med egna kommandon, så slår du snart SIMONSBASIC. □

FAM



behövt en förklaring.

Bokens del 2 avhandlar "mönster", vilket vi väl skulle kalla grafik. Kalejdoskopet är väl kanske den intressanta delen där. Hur man gör rektanglar kan vara nyttigt att veta, men inte spännande.

Bokens del 3 kallas hemundervisning och är en typisk amerikansk tillställning med Amerikas presidenter och stater. Detta kan ju översättas till kungar och landskap och varför inte en Eriksgata.

Program med grundläggande matematik ingår liksom stavningsprogram. De senare är nyttiga för praktiskt taget alla, vilka utsatts för den moderna s k undervisning, vilken serveras på våra läroanstalter. Ett program om mått och vikt måste naturligtvis göras om högst betydligt till europeisk standard.

Ett avslutande program för en minneslista är en nyttig och ofta förbisedd användning av hemdatorn.

Boken vore värd ett bättre öde än att överlämnas till den svenska läsaren oöversatt och obearbetad. Författarens ambitionsnivå slår inte igenom till efterföljande led, tyvärr.

Bra Basic-bok

Böcker om Basic är ungefär lika vanliga som sandkorn i Sahara och vanligen lika intressanta.

Den bok vi presenterar för läsaren är avsedd direkt för Commodore VIC 64, men även för VIC 20 samt datorerna i 3-, 4- och 8000-serien. Det föreligger trots allt en god kompabilitet för rena Basic-program.

Commodore 64 – Der Computer für Einsteiger und Aufsteiger av Hans Riedl och Franz Quinke, hft 156 ss+4. ISBN 3–470–80421–6.



Det blir lätt ett slags inavel i böcker att detta slag, då den ena författaren naturligtvis har läst alla de andra. Många program blir därför störande lika. Skönt är då, att författarna till denna bok endast har läst en engelsk bok, nämligen Commodores Programmer's Reference Guide.

Tyska läroböcker är ofta av mycket hög kvalitet och är i förhållande till svenska och amerikanska böcker skandalöst billiga.

Forts på sidan 42

Forts från sidan 41

"Bra Basic-bok"

Föreliggande bok: Commodore 64 – Der Computer für Einsteiger und Aufsteiger, Daten, Text, Grafik, Musik – är ett gott exempel på tysk bokkonst. Föredömligt kort – 149 sidor med tryck i rosa, grått och svart. Färgerna rosa och grått ligger över textfält av särskild betydelse, där något nytt presenteras eller där författarna vill repetera något.

Boken bygger på decimalsystemet, d v s kapitlen numreras från 1 och uppåt. Under huvudämnet 1. kommer så delar av huvudämnet och behandlas då under t ex 1.1. o s v.

Författarna behandlar inte ämnet renodlat utan ger även läsaren inblickar i utvecklingen av datorerna där nu mjukvaran tagit över huvudparten av försäljningen. De går igenom VICarnas användningsområde samt ger emellanåt några små nyttiga tips åt läsaren.

Sedan smyger man sig sakta på läsaren med små Basicsnuttar. Man visar i flödesschema hur man logiskt måste bygga upp ett program samt vilka Basicord man bör använda för att nå en lösning.

Nästan halva boken ägnas åt Simons Basic och ger exempel på grafik, text och musik. Alla exempel är korta men instruktiva och det är ju inte alltid ett långt program är bättre än ett kort.

En trevlig bok, vilken hindras i sin spridning av att den är skriven på tyska. □

RECENSIONER



Att göra om VIC 64 till 80 kolumner

COMPUTE'S Gazette for Commodore No. 9, September 1984

Ett av de stora slagnumren i Gazettens septemberutgåva är en artikel med åtföljande maskinkodat program som gör om VIC 64 från en 40 kolumners maskin till en 80 kolumners – d v s 80 tecken per rad. En vanlig TV har tyvärr inte den precision som krävs för att göra 80 tecken per rad klara och tydliga och författarna rekommenderar därför en videomonitor i detta sammanhang.

"Screen 80" utnyttjar en högupplösningsskärm som placerats i RAM som ligger under operativsystemet i ROM. Det är alltså möjligt att samtidigt lägga ut högupplösningsgrafik och texttecken i Screen 80. Man kan också använda redigeringsfunktionerna på 64an på ungefär det sätt man är van vid.

Om man inte vill använda de tecken som finns inbyggda i 64ans ROM finns det också en teckeneditor, "Custom -80", bifogad. De tecken som man med hjälp av joystick kan definiera i denna editor kan permanent läggas till Screen 80 vilket innebär att dina egna tecken alltid kan laddas in tillsammans med Screen 80 och Custom -80, som båda är maskinkodade, knappas bekvämt in med hjälp av COMPUTE's eget speciella hjälpprogram, MLX, som gör att risken för misstag minimeras.

SPEL

Avdelningen "games" innehåller i detta nummer två programlistningar, båda för såväl VIC 20 som VIC 64. Det ena spelet "Mystery at Marple Manor", är ett s k "adventure"-spel och det andra, "Treasure Hunt", är ett traditionellt joystickkontrollerat spel.

MYSTERY AT MARPLE MANOR

Handlingen utspelas i ett gammalt hus med 14 rum. Vissa rum är låsta, men det finns nycklar som man med tur och skicklighet kan hitta. I något av rummen finns ett lik och i något rum finns mordvapnet. I huset finns också ett antal misstänkta personer som kan ha utfört dådet. Din uppgift som detektiv är att avslöja mördaren, mordvapnet och det rum i vilket mordet begåtts.

En extra finess med det här spelet är att man kan vara flera spelare/detektiver vilket gör det möjligt att lura motspelare till felaktiga slutsatser. Sammanfattningsvis är detta en riktigt kul spelidé som manar till efter-

följd, kanske t o m en svensk sådan?

TREASURE HUNT

Strandsatt på en öde ö ska man samla ihop de skatter piraterna lämnat efter sig. Men piraternas osaliga andar vilar över platsen... Man styr sin skattsamlare med joystyck till skatterna medan de osaliga andar man ska akta sig för rör sig på ett mer slumpmässigt sätt, allt för att störa din framfart. Spelidén verkar vara ganska fräsch.

UTBILDNINGSHÖRNAN

Varje nummer av Gazetten innehåller en avdelning som kallas "Education/Home Applications". Där förekommer det ofta program för yngre datoranvändare. Ibland är det s k utbildningsspel som presenteras. I detta nummer får vi ett matematikprogram, "Learning to Count", som lär barn i förskoleåldern att räkna. Utbildningsprogram för barn/ungdom är "Big Business" i staterna just nu, men när får vi se roliga och vettiga utbildningsprogram anpassade för svenska förhållanden?

PROGRAMMERING?

"Disk Tricks" kan vara bra att kunna om man t ex skulle vilja återkalla diskfiler som av misstag blivit raderade. Här ges en grundlig genomgång av hur information lagras på dina disketter och hur man, på ett mer direkt sätt än vanligt, kan komma åt denna information. Om du är intresserad av dessa frågor rekommenderas att du först läser Jörgen Stiernborgs utmärkta artikel i VIC rapport nr 7, 1984, som behandlar i stort sett samma saker, fast på svenska.

Feta rubriker – "Screen Headliner" – heter ett klurigt program som kan användas för att få fyra gånger uppförstorade versioner av de tecken som redan finns eller kan skapas på VICarna. Det här är användbart om man t ex vill få lite annorlunda titelutskrifter från sina program.

Ytterligare en användbar rutin som man bjuder på i denna Gazette är "Cursor Get", en cursor i GET-sammanhang för att hålla reda på var på skärmen man stoppar in sin information.

I programmeringsavdelningen finns varje månad en spalt om maskinkodning för nybörjare. I detta nummer behandlas vilka krav man bör ställa på en bra assembler. Man ska kunna använda Commodore's redigeringsfunktioner, man ska kunna kommentera koden (motsvarande REM-satser), snabbt kunna assemblera till disk o s v. Vi får också lära oss skillnaden mellan källkod (source code) och objektkod (object code).

SKVALLRET – EDITOR'S NOTES

Inte minst viktig är ledarsidan. Här får vi veta att Commodore's f d chef, Jack Tramiel, köpt in sig i Atari. Ska Atari, som tydligen brottats med problem den senaste tiden, i och med Mr. Tramiel bli lika framgångsrikt som Commodore varit de senaste åren? □

G. Berglund

Septembers lottodragning

Denna månad var det familjen Sand som vann i VIC-rapports lottodragning bland våra prenumeranter. Programmet heter Genesis. Vi på redaktionen gratulerar.

Forts från sidan 29

”Att tjäna pengar”

Men det är precis vad dom är. Dom VILL ha tag på BRA program och gärna svenska sådana. Och är det så att dom skickar tillbaka programmet med ett meddelande om att det inte riktigt höll den klass av program dom är ute efter. Så vad gör det?

Datorn brinner inte upp för det (om nu inte du tänder eld på den). Och det här med att programmet är skrivet i BASIC behöver inte alls vara någon nackdel. Men det är klart om det är ett typ Action-Skjuta-Smasha-Spel bör det vara gjort i maskinkod. Jag hade som många andra gjort en variant av det populära spelet Q*BERT på min dator VIC 20.

TOG MOD TILL MIG

En dag när jag satt och tittade i en katalog från GRANA SOFTWARE så stod det på en sida att dom sökte efter program och uppmanade samtidigt att skicka in en kopia om man nu gjort bra program. Jag hade ju gjort ett! Men om det var tillräckligt bra visste jag inte men jag tog mod till mig och skickade en kassett med en kopia på spelet samt en beskrivning hur det hela fungerade.

ETT TJOCKT KUVERT

Det är väldigt bra om man skickar med en beskrivning av programmet, gärna detaljerat. Efter ca tre veckor fick jag ett tjockt kuvert och då jag såg att det var min kassett tänkte jag – Nähä det dög inte. Men där tog jag miste för med i kuvertet var en liten lapp från en kille som heter Peter Tegelberg och han skrev att dom ville köpa rätten till spelet! Och ni kan gissa vem som blev glad. Understrykas bör alltså att hela programmet var skrivet i BASIC och på en oexpanderad maskin. Förresten så heter spelet QU-BERT.

Lycka till!

Patrik Holmström, Westerås VIC Klubb

VICtig information

VIC rapports databas har nu kommit igång ordentligt. För dig som inte har provat den ännu kan vi berätta att följande utrustning krävs: Förutom en VIC behöver du ett 75/1200 modem samt ett teledataprogram. Därefter ringer du upp basen på nummer 08-19 06 16. Basen frågar då efter ett kodord vilket är 90000VIC64. Observera att shifttangenten ska hållas nedtryckt när du skriver VIC.

Du får inte gå ur basen utan att meddela den att du gör det. För annars spärrar du ingången för andra användare. Detta är inte så farligt om det sker under normal arbetstid eftersom vi på redaktionen då kan bevaka basen.

Däremot är det mindre roligt om det sker på kvällstid. Tänk på alla andra som är intresserade av att använda basen. □

Forts från sidan 25

”Drink-bar, VIC 20”

```
430 PRINT"              /‾‾‾‾‾‾\"
440 PRINT"         L    __|"
450 PRINT"        /        \|"
460 PRINT"       |          |_"
470 PRINT"        \_        |\"
480 PRINT"          \______/|"
490 PRINT"MOLNIGT !!!!":FORI=1TO4000:NEXT:GOTO590
500 PRINT"                 |"
510 PRINT"          |"
520 PRINT"        /‾‾‾\   __"
530 PRINT"   --  |     | "
540 PRINT"        \___/  "
550 PRINT"          |"
560 PRINT"          |"
570 PRINT"       HETT!!!"
580 FORI=1TO4000:NEXT
590 C=2.5*A
600 POKE36879,25
610 D=A*B
620 IFJ=1THENK=INT(RND(1)*30)+1:GOTO650
630 IFJ=2THENK=INT(RND(1)*70)+30:GOTO650
640 IFJ=3THENK=INT(RND(1)*60)+100
650 IFD<KTHENANT=A:GOTO670
660 ANT=INT(K/B)
670 IFB>4.5THENANT=INT(ANT+2.5-B):IFANT<0THENANT=0
680 PRINT"";ANT;"ST AV";A;"
690 PRINT" BLEV SALDA"
700 VIN=(ANT*B)-C
710 PRINT"VINST";VIN;"KRONOR"
```

```
720 SS=INT(SS+VIN)
730 PRINT"SUMMA";SS
740 PRINT"";W;"OMGANGAR KVAR"
750 IFW=0THEN800
760 W=W-1
770 PRINT:PRINT"TRYCK VALFRI TANGENT!"
780 GETA$:IFA$=""THEN780
790 GOTO160
800 IFHS<SSTHENHS=SS
810 PRINT"SLUT HIGH SCORE";HS:GOTO850
820 GOTO800
830 DATA 225,300,225,360,225,240,228,120,231,360,231,240,228,120,231,240,232,12
,235
840 DATA 720,240,360,235,360,231,360,225,360,235,240,232,120,231,240,228,120,22
,480
850 PRINT:PRINT"TRYCK VALFRI TANGENT!"
860 GETA$:IFA$=""THEN860
870 SS=0:GOTO60

READY.
```

GRATISANNONSER

SÖKES

Brevvänner sökes
Jag är en VIC 20 ägare som vill byta program och erfarenheter med andra VIC 20 ägare. Skriv till Anders Kuhmunen, Box 94, 960 40 Jokkmokk.

Brevvän sökes VIC 64
Söker brevvän med VIC 64. Ganska ny på området. För byte av program m m. Per-Olof Karlsson, Ringvägen 3A, 682 00 Filipstad. Vill också köpa följande spel: Zaxxon och Worms, endast kassett. Ring 0590/129 25 efter 20. Peo.

KÖPES

Köpes
Jag köper begagnad VC 1541 Floppy Disk. Ring till: Hans Lindkvist, tel 0911/100 02.

Köpes
Zaxxon eller Dallas Quest för högst 50 kr/st. Ring 0415/105 84 Mats.

Köpes
Printer 132-teckens med Centronics-snitt, Modem 300 och 1200, nakna Diskdrivar, engelska elektor, practical electronics, amatörradio-sändare. Tel. 046/13 11 51.

VIC 64
Jag vill köpa Sprite Aid (begagnat) för max 50 kr. OBS! Jag vill ha instruktionsboken med. Birger Wranå, Movägen 6, 352 49 Växjö.

Köpes
Skrivare till VIC 64. Tel. 060/15 92 87.

Köpes
Instruktionsbok till spelet M.U.L.E, 031/43 14 30 Kaj (kvällstid).

VIC 64 program köpes
Text 64, Calc Result, Scope 64, Magpie, Practifile, Vip Terminal, Buscard, B1–80 Column Adapter, Color 80, Lättfil och andra högkvalitativa nyttoprogram köpes för rätt pris. Tel. 0171/309 36 Petter.

VIC NEWS (nytt) köpes
Vill köpa VIC NEWS 1-årg 1 och VIC NYTT 2-årg 2 i gott skick. Säljer VIC rapport (nr/årg) 5–6/2, 1/3, 2/3, 3/4, alla i nyskick, för 15 kr. inkl porto. Calle Einarsson, Kaprifolvägen 32, 453 00 Lysekil.

Hjälp önskas med ett Bowlingprg. VIC 64
Finns det någon som kan hjälpa mig med ett prg. till ca 300 starter på disk, samt utskrift till skrivare i alla klasser som sorterar efter högsta resultatet. Samt ett statistikprg. för en liten Bowlingklubb (50 medl.) OBS! Det är bråttom! Är där någon som redan har ett sådant så hör av dig till mig. Sänd ditt svar till Leo Dupont, Smaragdgatan 15, 267 00 Bjuv, tel. 042/825 44.

Köpes
VIC rapport årg 1 nr 1 och årg 2 nr 2. Gunnar Lindström tel. 0753/372 96.

BYTES

Spel VIC 64 bytes
Ring of Power, The Hidden Treasure of Inca, Det stora Nordiska Kriget, Cirkus, The Time Machine, Hungry Horace, Super Dogfight. Bytes mot äventyrsspel till VIC 64. Telefon 0591/136 76 Ove.

VIC 20 program bytes
VIC 20 maskinkodsspel och nyttoprogram bytes. Erbjudande: Du får lika många program av mej som jag får av dej + 2 extra. Tel. 0411/260 12 Magnus.

VIC 64 program bytes
Jag önskar byta spel och nyttoprogram

Forts "Bytes"

till 64an ev. säljes billigt för 25 kr/st. Ulf Eriksson, Hagmarksvägen 22, 824 00 Hudiksvall. Tel. 0650/157 82.

VIC 64 spel bytes
Jag har: Fort Apocalypse, Escape Mcp, Panic 64, Scramble, Spaceinvaders, Pole Position, Pilot 64. Bytes mot: Clown 64, Omega Race 64, Sea Battle 64, Night Driver 64. Ring 0755/188 49 efter 18.

VIC 64
Spel bytes ev säljes. Superbra spel som t ex Jumpman Jr, Fort Apocalypse, Dig Dug, Pole Position, Zeppelin, Pharaoh's Curse. Ring 08/85 67 37 fråga efter Micke. OBS! Endast på band. PS. Vill gärna byta till mig Congo Bongo, Beach Head.

VIC 20 spel bytes
Skicka lista på vilka spel du har så skickar jag en lista tillbaka inom 2 dagar... Per Öberg, Möllevägen 24C, 236 00 Höllviken. Tel. 040/45 44 20.

64 program bytes
Spel och nyttoprogram till CBM 64 bytes, ev säljes. Endast på disk. Tel. 0758/365 80 Christofer.

Bytes VIC 64
Jag byter bort följande spel: Fort Apocalypse, Snokie, Cup Final eller Jucie mot Archon eller M.U.L.E. OBS! Alla spel på disk! Ring Charlie tel. 0764/319 62.

VIC 64 spel bytes
64-spel bytes (helst på disk) ring: 0760/137 06 John.

VIC 64 – VIC 20
Jag byter och köper bra program till VIC 64 på disk. säljer även Metagalactic Llamas och Skyhawk till VIC 20 (original). Ring Jonas 0411/230 40.

VIC
VIC 20, 64 spel bytes, ev säljes. Skicka lista till Mikael Håkansson, Västratornsvägen 5, 222 50 Lund.

VIC 64
Jag byter spel med dig. Jag har t ex Cupfinal, Pole Position, Jumpman Jr, The Hobbit + 25 andra på kassett. Söker Flight Simulator på kassett. Staffan Polerius, Vårlöksvägen 10, 434 00 Kungsbacka. Tel. 0300/154 01. Ring fort!

VIC 64 program bytes
Vi är två killar som vill byta program och erfarenhet, helst maskinkodsprogram. Tel. 054/18 11 09 Jonas.

COMMODORE 64 bytes
Program till CBM bytes. Spel och nyttoprogram bytes, ev säljes. Stor programpool. Tel. 054/18 11 09 Jonas.

64 spel bytes
Jag är intresserad av att byta M/C program med dig! Jag har allt från nytto – spel, mycket stort urval kas. eller disk. ring idag! Ulf 031/81 13 33.

Bytes
Scott Adams Adventureland & Pirate Cove mot spel i samma prisklass. Ring 013/15 10 17, Fredrik.

SÄLJES

Dammskydd
VIC 20/64 + bandspelare, 40 kr. Disk 1541, 30 kr. Finns även för annan utrustning. Specialskydd kan beställas. Tel. 0303/776 15 Lars Olsson.

Printerpapper
I stående A4 säljes för endast 50 kr/500 ark. Ring 08/87 99 20 Janne.

VIC 20 säljes
VIC 20, bandspelare, expansionskort 4 portar, 16K RAMminne, superexpander, böcker och spel. Nypris ca 5 500 kr. Säljes i ett paket eller i delar, till ett billigt pris. Ring 0481/128 24.

Bytes VIC 20
Expansionsenhet med 8K och ingång för 3 st cartridge bytes mot bandstation endast skriftligt svar: I Marko, Box 1104, 500 11 Borås.

Säljes till VIC 20
Expansionsenhet 8K med ingång för 3 st cartridge 350 kr samma med två ingångar 150 kr obruten. Introduktion till Basic 2 150 kr obruten från spel till Basic 150 kr resp allt för 900 kr. Endast skriftliga svar Ilja Marko, Box 1104, 500 11 Borås.

Modemprogram VIC 64
Terminalemuleringsprogram. Enkla 150 kr, supersmarta som kan allt 350 kr. Menystyrda 50 – 2400 baud. Spara på disk skriv ut m m, även för tape. Konverterar textfiler till Basic. Utförlig manual. Ring 0760/517 87 för vidare info. Krister Kristiansson.

Se hit alla spelgalningar
Välkända VIC 20 spel säljes mycket billigt. Ring 0411/260 12 Magnus.

Säljes
Cartridge till VIC 20: Star Battle, Game and watch. Bordsspel: Donkey Kong Jr pris 100 kr per styck, porto tillkommer. Skriv eller ring till Erik Bertell, Ringv. 23, 824 00 Hudiksvall. Tel. 0650/161 74.

VIC 20 säljes
VIC 20 med tillbehör obetydligt använd. 3/8/16K minne + expansionsenhet + superexpander + manual + 4 böcker + 20 tal spel och nyttoprogram + bandspelarinterface. Köpt jul -83. Allt för endast 1900 kr. 0451/815 76 Hans.

VIC 64 snabbladdare
Svensktillverkat snabbladdarprogram som gör att kassettprogrammen sparas och laddas 8–10 gånger snabbare! Pris 60 kr. Tel. 0451/815 76 Hans.

VIC 64 program
Mina spel och nyttoprogram utförsäljer jag billigt. 3 st för 100 kr, ex Supergridder, Lazerzone, Fotball, Le Mans register och många fler. 0451/815 76 Hans.

Data – Stryktips
Datorprogram för de populära systemen U, UM, S och SM säljes. Programmen, som är anpassade till de nya kupongerna sparar dina favoritnycklar, bildar och skriver ut aktuella rader samt rättar. Programmen passar VIC 64/exp, VIC 20 och är enkla att använda. Pris: 100 kr/program, 300 kr för samtliga fyra. Mats Olsson, tel. 046/13 20 53.

Säljes
VIC 64 program säljes och bytes. Tel. 08/37 14 56.

VIC 64/20
Skona din VIC använd RESETknapp för att tömma datorn. Endast 45 kr. Ringer du mycket? Låt datorn sköta jobbet. Telefonringare + program 97 kr. Autofire i proffsutförande 95 kr. Skriv till Niklas Björklund, Bråviksv. 18, 603 65 Norrköping.

Blue Max
Hör upp alla VIC 64 ägare! Jag säljer mitt häftiga spel Blue Max för 200 kr. Släng dig fort på telefonen! Martin Augustsson, tel. 031/29 97 27.

Dubbelfloppy VIC 64
MSD-Diskdrivar passar till VIC 20 och 64 + PET 4000-, 8000-serier. 1 st Dubbeldrive 8 500 kr, 1 st enkeldrive 4 500 kr, 1 st Superbox 64 700 kr., 1 cardco + G printerinterface 650 kr. Ring Göran tel. 044/11 67 96 säkrast kvällstid.

Rankning av travhästar VIC 64
Registerhanteringsprogram där du får singeltips, systemförslag, outsidetips. Du kan avbryta, spara, hämta. Möjlighet att lägga in egna värderingar. Utförlig dokumentation/manual. Ronny Larsson 0383/158 96, 135 kr.

Modem till Commodore 64
75/1200 och 300/300 Baud 1 000 kr. Helt nytt. Ej inkopplat. Ring 021/33 27 30 efter kl 17, Mats.

Scott Adams VIC spel
Lösningar och ledtrådar till alla fem. Säljes för 10 resp 5 kr. ring 013/15 10 17, Fredrik.

VIC 64 säljes
64 + Philips Computor monitor + bandspelare + Simons Basic + 6 handböcker. 3 mån. garanti kvar. 6 300 kr. Tel. 0413/226 29 eller 232 56, fråga efter Björn.

VIC 20 säljes
Dator, bandspelare, joystick, 16K, Butiplus (hjälpkommando + M kodsmon + 3K), motherboard, böcker och massor av program säljes för 2 200 kr. Ring Martin Carlsson. Tel. 0302/412 04.

Simons program VIC 64
Skicka namn och svarsporto till mig så får du en lista på mina läckra program. Johan Harrysson, Båtv. 5, 590 61 Vreta Kloster.

VIC 20
Slotcartridge endast 50 kr. Magnus tel. 040/46 48 46.

VIC 20 paket
VIC 20 dator + bandstation + Superexpander + 6 st spelcartridge + joystick + RS232C Terminal Type + Handbok + Programhandbok + 6 st böcker + glosförhör + massor av spel. Har kostat 7 100 kr, allt för endast 3 500 kr. Pris kan diskuteras. Magnus, tel 040/46 48 46.

VIC 20 plug in
Jag säljer Jelly Monsters cartridgen till VIC 20 för endast 125 kr (nypris 298 kr) Lyft luren slå 033/856 32, be att få tala med Björn!

VIC 64 program
Nyttoprogram och spel till 64an bytes, säljes billigt. Tel. 0650/231 17 Jonas.

VIC 20 spel säljes
3 st för 50 kr, arrow, catacombs + 3K, Race + Rack, Amok (JS), Grottbasen, Minefield, Arrow, Luffarschack + 3K, Logger + 3K, Lotto med rättning + 3K, Ladders + 8K, Bandit. Vilka vill du ha? Tore Gullstrand. Tel. 0764/603 25.

VIC 64 program
Spel och nyttoprogram säljes, pris från 15–50 kr, inga listskydd. Beställ superspelet 2002 + en katalog för endast 40 kr! Skriv eller ring till Göran Sundin, Kackelabacken 30, 871 00 Härnösand. Tel. 0611/104 72.

VIC 20 tillbehör
VIC 1111 16K RAM minnesexpansion 300 kr. VIC 20/672 expansionsenhet för 3 st. Cartridges 200 kr. Per Allan Persson, 042/22 21 15.

Program VIC 64
Nytto och spel till 64an säljes billigt eller bytes på disk. Tel. 0758/423 88 Jonas.

VIC 20 program säljes
Ordbehandling med manual + 16K för 140 kr. VIC tape backup kopierar dina bästa program och ha i reserv, går även på skyddade program, 70 kr. Tore Gullstrand, Norrgårdsv. 127, 184 00 Åkersberga.

VIC 20 spel
Mine-Madness från Thorn-Emi säljes för endast 350 kr. Har kostat 450 kr. Ring eller skriv till Kristoffer Hansson, Bogesundsg. 30, 252 00 Helsingborg, tel. 042/29 66 26.

VIC 20 spel säljes
Spel som Donkey Kong, Anti Matter Splatter, Mastermind, VIC Man, Jagad, Störtlopp och Asteroids för endast 100 kr. Skicka 1 band + pengar till Eric Jeising, Aspg. 1, 565 00 Mullsjö.

VIC 20 med Å Ä Ö säljes
VIC 20 ÅÄÖ-kit, superexpander, resetknapp, 2 program, handböcker, VIC Revealed. 2 000 kr. Ev säljes 64 K RAM. 650 kr. Magnus Franzen 0331/12 02 85.

VIC 20 säljes
VIC 20 + kassettminne + 3 böcker. Superexpander, Programmers Aid, 3 port expansionsenhet, över 200 program, spelmoduler, Paddles m m. Nypris ca 7 000 säljes för 4 000 kr. Tel. 0498/172 91, Svante Larsson.

Summer Games till
VIC 64 för 100 kr. Ring Andre 08/85 66 49.

Databas CBM 64
Seriös användarvänlig registerhantering med diskett. Rutiner för uppdatering, A4/etiketter-utskrifter, kassettbackup etc. Diskett + Manual 255 kr. Tel. 0755/135 02 Kent.

Vi bjuder! Till VIC 64
har du hört: prova programmen först, betala sedan! "Ring för Basic" ger 100% garanti mot besvikelse. Nya intressanta program ger oss trygghet. 10 program – 1 per mån pris: 65 kr. Skriv el ring I Östlund, Jöns Persv. 9, 818 00 Valbo, tel. 026/13 04 34.

Till VIC 20 säljes
32K expansion, superexpander M/3K, Mothercard M/3 anslutningar, Creaphic, programmers AID, Multisound – synthesizer egna tecken, Höggrafikpack. Nypris 3 800, säljes samlat för 1 900 kr. Tel. 08/761 20 21, Erik Sölving.

VIC 20 säljes
VIC 20 + 16K bytes + maskinkodsmonitor + bandspelare + 9 extra kommandon + 2 spel böcker och 4 manualer + ca 300 program, bl a Hopper, anihilator, VIC-men etc. för 3 000 eller högstbjudande. Nypris 7 000. Tel. 031/45 63 34, Mikael.

VIC rapport nummer 10 utkommer den 12 november

Ska du köpa en joystick? Vänta då tills den 12 november för då presenterar VIC rapport inte mindre än 10 stycken olika joysticks som du kan hitta på den svenska marknaden.

Söker du jobb?

Inget problem om du har lite datorkunskap i ryggen. Men hur fungerar svensk utbildning inom dataområdet? Läs och lär.

Tävling

Presentation av alla våra 2a-pristagare i tävlingen. Kanske just du har vunnit!

Massor av program och spelrecensioner.

Handic Press
Box 42054, 126 12 Stockholm Tel. 08-744 59 20

MASSBREV

